VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro vytváření a prezentaci vzdělávacího obsahu v mobilních zařízeních

Tool for Education Material Presenting on Mobile Devices

2014 Bc. Veronika Žółtá

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Veronika Žółtá**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Nástroj pro vytváření a prezentaci vzdělávacího obsahu v mobilních zarízeních
Tool for Education Material Presenting on Mobile Devices

Zásady pro vypracování:

Cílem této práce je navrhnout nástroj pro vytváření e-learningových kurzů spustitelných v mobilním zařízení. Součástí práce je implementace a reálné nasazení tohoto nástroje v prostředí internetu spolu s mobilní aplikací, která bude sloužit k zobrazování a interakci se vzdělávacím obsahem.

1. Nastudujte problematiku elektronického vzdělávání, se zaměřením na vytvoření univerzální struktury pro vzdělávací materiál spustitelné na mobilních zařízeních.
2. Zvolte vhodnou technologii pro tvorbu on-line nástroje, který bude představovat obslužný prvek k vytváření a správě multimediálního vzdělávacího obsahu.
3. Obsah materiálu bude rozdělen do kurzů, jejichž struktura bude hierarchicky rozvrstvena od těch největších nezávislých celků po ty nejmenší (výukové objekty).
4. Seznamte se s problematikou implementace aplikací pro mobilní zařízení (na platformách Android,Windows Mobile, iPhone) a zvolte vhodnou platformu, která bude pokrývat problematiku doručování a testování vzdělávacího obsahu se zpětnovazebními prvky.
5. Na závěr proveďte reálné nasazení a ověření funkčnosti výsledného řešení.
6. Uveďte další možné kroky k rozšíření stávajícího řešení.

Seznam doporučené odborné literatury:

Průvodce programováním mobilních aplikací od Mark L.Murphy a vydavatelství Computer Press, Brno, 2011

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radoslav Fasuga, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne: *29. dubna 2014*

V. Žoldáková
.....
podpis studenta

Poděkování

Ráda bych poděkovala vedoucímu práce Ing. Radoslavu Fasugovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Diplomová práce se zabývá tvorbou výukového nástroje určeného pro e-learning. V teoretické části jsou objasněny základní typy e-learningových systémů a jejich konkrétní existující řešení v podobě nástroje Adobe Captivate nebo Moodle. V praktické části se práce věnuje návrhu a implementaci vlastního nástroje, který řeší jak problematiku vytváření a uspořádání kurzů, tak i problematiku testování. Funkce testování bude navíc dostupná i z mobilních zařízení, pro které bude systém optimalizován.

Klíčová slova

e-learning, výukový nástroj, problematika testování, adaptace, uspořádání kurzu, LMS, Adobe Captivate, Moodle, on-line výuka, responzivní design

Abstract

This thesis deals with the creation of educational tools designed for e-learning. In the theoretical part is explained the basic types of e-learning systems and their particular existing solutions as Adobe Captivate and Moodle. The practical part of the thesis is about proposal and implementation of learning system which solves the issues of creation and organization of courses, as well as the issue of testing. Function of testing will also be available from mobile devices, for which the system will be optimized.

Key words

e-learning, learning tool, the issue of testing, adaptation, course structure, LMS, Adobe Captivate, Moodle, online learning, responsive design

Seznam použitých zkratek

Zkratka	Význam
LMS	Learning management system
LCMS	Learning management content system
HTML	HyperText Markup Language
SCORM	Shareable Content Object Reference Model
ADL	Advanced Distributed Learning Initiative
AICC	Aviation Industry Computer-Based Training Committee
IMS	IP Multimedia Subsystem
IEEE	Institute of Electrical and Electronics Engineers
SCO	Sharable content object
API	Application Programming Interface
XML	Extensible Markup Language
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
UML	Unified Modeling Language

Obsah

1 Úvod

E-learning se dnes běžně využívá na mnohých školách i korporacích. Může mít různou podobu. Od jednoduchých úložišť vzdělávacího obsahu až po sofistikované systémy, které díky svým nástrojům umožnují téměř plnohodnotnou výuku na dálku. Mají výhodu v tom, že dokáží studentům ušetřit čas strávený na cestách za vzděláním a lektorovi částečně odpadá opravování testů, jejichž vyhodnocení lze zautomatizovat.

S elektronickým vzděláváním se však pojí několik problematik. Jak přenášet a formulovat různorodý výukový materiál do elektronické podoby. Jak obsáhnout, co nejširší potřeby lektorů při vytváření kurzů, a jak výuku personalizovat a přizpůsobit ji potřebám každého žáka. Cílem diplomové práce je navrhnout a naimplementovat systém, který by řešil všechny zmíněné problematiky a nabízel univerzální nástroj pro vytváření, správu a prezentaci výukových materiálů a to i prostřednictvím mobilních zařízení.

V první teoretické části práce jsou objasněny základní typy e-learningových systémů a rozebrány konkrétní existující řešení v podobě nástroje Adobe Captivate nebo Moodle. Druhá praktická část práce se věnuje návrhu a implementaci vlastního nástroje, který umožnuje testování a vytváření univerzálních struktur kurzů. Poslední část práce se zabývá optimalizací systému pro mobilní platformy a možnostmi, jakými může být systém v budoucnu rozšířen.

Diplomová práce navazuje na bakalářskou práci, jejímž tématem byla tvorba nástroje pro sebetestování se speciálními typy úloh. Původní řešení se hlavně zabývalo problematikou, jak výukový materiál zaobalit do různých typů elektronických úloh. Nynější řešení využívá získaných poznatků, a na jejich základech staví nový systém, který navíc řeší problematiku návrhu a tvorby komplexních kurzů a jejich různých způsobů procházení.

2 Problematika elektronického vzdělávání

Za účelem elektronického vzdělávání vznikla celá řada výukových systémů. Hlavním důvodem pro vznik e-learningu bylo, aby výuka mohla probíhat na dálku, a tím se ušetřil čas a finanční prostředky spojené s cestováním.

Kurzy se zprvu vytvářely ručně a prezentovaly prostřednictvím propojených webových stránek. Nastával zde ale problém, že tutoři tyto kurzy nedokázali sami vytvořit bez znalostí jazyka HTML a CSS. Navíc e-learning vyžadoval vyšší stupeň interaktivity a funkčnosti než nabízely statické stránky (testování, sledování výsledků studenta, řízení výuky). Brzy proto vznikly různé intuitivní nástroje pro vytváření vzdělávacího materiálu a řízení výuky zvané **Learning Management system** označované zkratkou LMS.

2.1 Typy e-learningových systémů

E-learningové systémy jsou dnes poměrně dost využívané vzdělávací řešení, které umožňují vzdělávat zaměstnance firem nebo studenty na dálku. Používají je zejména velké firemní korporace, školní instituce a jazykové centra. Většina e-learningových systémů má formu webových aplikací a jsou dostupné on-line. Výuka díky internetu může probíhat kdekoliv a odkudkoliv. [2]

Podle svého zaměření se dělí se do dvou hlavních skupin:

- **Learning Management system (LMS)** – nástroj určený pro spouštění vzdělávacího obsahu a k řízení výuky
- **Learning Content Management system (LCMS)** – autorský nástroj k tvorbě kurzů a jejich publikování

2.1.1 Systém pro řízení výuky (LMS)

LMS jsou primárně určeny pro organizaci výuky, testování žáků a sledování jejich pokroků. S LMS nejčastěji pracují lektoři, kteří skrz systém zadávají svým studentům různé úlohy a řídí jejich výuku na dálku. U distančního studia, kdy student nemůže školu každodenně navštěvovat je toto řešení jedinou možností, jedná se o tzv. Blended learning.

Funkce LMS jsou přizpůsobeny potřebám lektorů, aby dokázali zjistit znalosti studentů, a aby jim poskytli potřebné studijní materiály a informace týkající se organizace výuky. LMS běžně obsahují tyto funkcionality:

- Evidence a administrace uživatelů a jejich přístupových práv
- Centrální úložiště pro výukové materiály - elektronické kurzy, skripta, videokonference

- Nástroj pro spouštění výukového obsahu
- Sledování výsledků studentů a jejich aktivit
- Nástroje pro organizaci a plánování výuky - harmonogramy, nástěnky
- Komunikační nástroje mezi lektorem a žákem – chat, fórum

U LMS je hlavním požadavkem jejich otevřenost a schopnost začlenit do svého katalogu i výukové materiály vytvořené v jiných systémech od jiných škol. Proto často podporují standarty SCORM, AICC, IMS a jiné. [2]

Příklady dnes nejpoužívanějších LMS:

- **Blackboard** – celosvětově nejrozšířenější
- **Moodle** - open-source řešení často používané na českých vysokých školách
- **Desire2learn** [3]

2.1.2 LCMS

LCMS (Learning Content Management System) je nástroj určen k sestavování výukového obsahu didaktickým zpracováním zdrojových multimédií (textů, obrázků). Autorům umožňuje vytvářet komplexní výukové jednotky a kurzy, které se mohou následně publikovat do různých standardizovaných formátů (HTML, SCORM, swf), tak aby jejich obsah byl spustitelný samostatně nebo za použití LMS.

LCMS se nabízí tyto základní funkcionality:

- Nástroj pro vytváření a správu digitálního výukového obsahu
- Správu katalogu s již vytvořenými výukovými objekty
- Zajištění indexace a znuvupoužitelnosti vytvořených výukových materiálů, umožňuje vytváření tzv. question bank, které jsou úložištěm pro testovací otázky
- Podpora výukových strategií
- Prostředky pro individuální nastavení kurzu
- Rozhraní pro definování organizace kurzu (určení pořadí a hierarchie výukových objektů)
- Sledování aktivit uživatele a nastavení podmíněného vykonávání aktivit v rámci kurzu
- Publikování výukového obsahu

Tyto funkcionality jsou založené na nejčastějších požadavcích lektorů při vytváření kurzu. Často se výukový obsah vytváří pomocí LCMS a poté se za použití SCORM standartu pošle do LMS, kde je obsah prezentován a použit k výuce. Mnoho výrobců se snaží do jednoho produktu zahrnout obě varianty, čímž dokážou docílit vyšší integrace a lepších výhod než jaké jim nabízí SCORM. [1]

K LCMS patří například produkty jako *Adobe Captivate* – desktopový nástroj od společnosti Adobe nebo *Kontis iTutor LCMS*. [2]

2.2 Struktura vzdělávacího obsahu

Vzdělávací materiál může být různorodý. Může se skládat z výkladových částí, interaktivních testů, grafů, map, videí a jiných částí. Vzniká zde menší problém jak tyto různorodé formáty zakomponovat do jednotné univerzální struktury, která by byla spustitelná na jakémkoliv zařízení.

Touto problematikou se už delší dobu zabývá americká iniciativa ADL (Advanced Distributed Learning Initiative), která za účelem vytvoření jednotného standartu pro přenos výukových objektů vytvořila referenční model SCORM (*Sharable Content Object Reference model*).

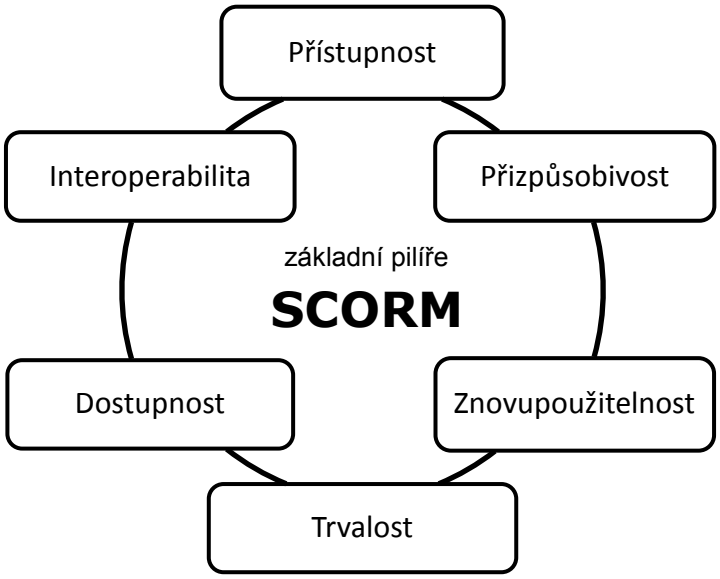
2.2.1 SCORM

SCORM neboli ***referenční model sdíleného obsahu*** je jeden z nejpoužívanějších standartů e-learningu. Je souborem specifikací a pravidel určujících jednotný model pro vytváření a distribuci vzdělávacího obsahu. Obsah vytvořený v souladu se SCORM specifikací je tak možno spustit a použít v jakémkoliv jiném systému (LMS) podporujícím SCORM. Právě tato přenositelnost je hlavním důvodem pro vznik tohoto modelu.

SCORM byl navržen na základě dřívějších specifikací pro e-learning vytvořených konsorcii jako je IMS Global Learning Consortium, Aviation Industry Network (AICC), Institute for Electrical and Electronics Engineers (IEEE) a aliancí Adriane. Všechny tyto specifikace přispěly k vytvoření jednoho modelu, jehož snahou je pokrýt všechna specifika e-learnignových systémů a potřeb jejich uživatelů. [4]

Zásady e-learningu

Při tvorbě SCORM standartu se vycházelo z následujících zásad a vlastností e-learnigu:



Obrázek 2.1: *Základní pilíře standartu SCORM* [18]

- **Přístupnost (Accessibility)** - schopnost nalézt a přistupovat k výukovým materiálům uloženým na jednom místě z jiných nezávislých umístění. Přístupnosti lze docílit použitím internetových technologií.
- **Přizpůsobivost (Adaptability)** - schopnost upravovat jednotlivé komponenty individuálním potřebám.
- **Znovupoužitelnost (Reusability)** - schopnost flexibility při začleňování výukových komponent do jiných kurzů. Vzdělávací materiál by měl být rozdělen do samostatných celků, které jsou nezávislé na kontexu a je možné je znovupoužít i v jiných souvislostech.
- **Trvalost (Durability)** – schopnost snášet přicházející technologický rozvoj bez nutnosti přepisování kódu a změny konfigurace. Obsah systému by měl být nezávislý na verzi softwaru.
- **Cenová dostupnost (Affordability)** – schopnost snížit náklady na výuku způsobem distribuce po internetu. Také znovupoužitelnost výukového materiálu šetří čas i finance spojené s vytvářením nových materiálů pro jiné systémy a účely.
- **Interoperabilita (Interoperalibity)** – je schopnost zachování univerzálnosti. Výuková komponenta sestavená jedním SW nástrojem by měla být spustitelná i v jiném rozdílném systému, než ve které byl vyvinut. [5][18]

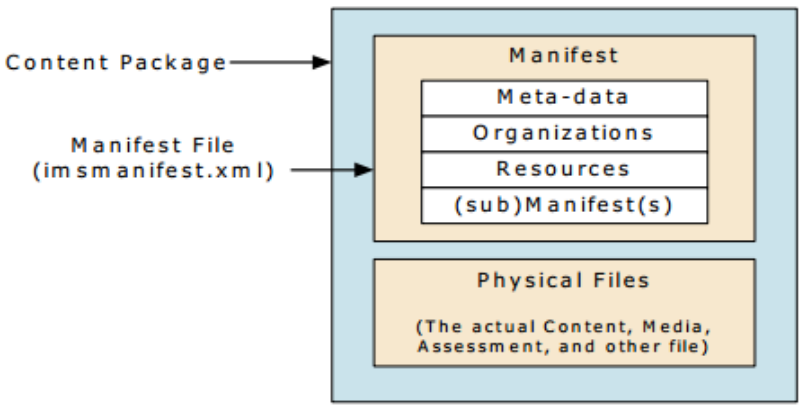
Struktura SCORM

Výukový obsah je podle specifikace SCORM začleněn do samostatných výukových objektů nazývaných *Sharable content object* zkráceně **SCO**. Tyto objekty se skládají z kolekce e-learningových zdrojových dat označovaných jako **assety** *(assets)*. Assety představující jednotlivé texty obrázky a grafy, jejichž spojováním dohromady vyučující vytváří komplexnější vzdělávací objekty.

SCO je nezávislou jednotkou výuky, která může být použita v jakémkoliv jiném kurzu a kontextu. Hlavním záměrem je dodržet jejich znuvupoužitelnost, a tím ušetřit čas a finance spojené s opětovným vyvářením stejných výukových materiálů pro jiné účely. SCO reprezentuje nejnižší vrstvu samostatně spustitelných zdrojů, které mohou komunikovat s LMS. Pro komunikaci používají jednoduché API metody pro inicializaci a ukončení objektu, díky kterým může LMS sledovat postup studenta v rámci kurzu. Každý objekt se může skládat z jiných objektů, a tím vytvářet stromovou strukturu celého kurzu. Hierarchie SCO je popsána v XML manifestu, kde jsou i všechny potřebné informace potřebné k jejich spuštění.

Struktura SCORM se skládá ze tří klíčových specifikací a to:

- **Content Packaging** – definuje, jak má být výukový obsah zabalen a poslán. Specifikace je založena na XML manifestu, který obsahuje všechny potřebné informace ke spuštění. Manifest uchovává informace o organizaci kurzu - jak je obsah členěn na jednotlivé SCO a jak lze každý z těchto objektů spustit. Do tohoto manifestu se zapisují i metadata, které popisují všechny komponenty obsahového modelu SCORM (assety, SCO, aktivity). Na následujícím obrázku 2.2 lze vidět uspořádání celého balíku Content Package. Kromě manifestu obsahuje balík i samotné fyzické soubory, které jsou v kurzu zahrnuty (multimédia, obrázky, texty, úlohy, apod.)



Obrázek 2.2: *Obsah Content Package [5]*

- **Run-time** – definuje běhové prostředí pro spouštění výukového obsahu, kterým je v tomto případě webový prohlížeč. Určuje způsob komunikace mezi LMS a doručovaným obsahem.
- **Sequencing** – specifikuje soubor pravidel, kterými lektor naviguje studenta skrz kurzem (mezi jednotlivými SCO). Obsahuje strom aktivit a informace o sekvenčním zpracování jednotlivých obsahových komponent. Nejtypičtějším příkladem podmíněné sekvence je závislost spuštění následující aktivity při splnění aktivity předchozí. Sekvenční pravidla jsou opět reprezentována XML dokumentem.

Celý manifest spolu s výukovými objekty je zabalen do jednoho samostatného výukového balíčku, který lze spustit na jakémkoliv LMS podporujícím SCORM. [5][4][6]

Využití v praxi

Specifikace SCORM umožnuje vytvářet plnohodnotné kurzy nezávislé na platformě. S použitím jazyků HTML, XML a Javascript je možno spouštět všechen výukový obsah pomocí webových prohlížečů na straně klienta. U většiny velkých e-learningových systémů je tento standart podporován a využíván pro import a export výukových balíků z jiných LMS.

Umožňuje sledování výkonu studenta – které objekty dokončil, úspěšně vykonal a za jaký čas. Neumožnuje ale vracet více výsledků pro jeden objekt. Stejně jako většina standardů i tento nenabízí neomezené možnosti, ale každým rokem se mění a vylepšuje.

Ačkoliv je dnes SCORM brán, jako hlavní standard pro strukturu a přenos vzdělávacích materiálů, mnoho systémů preferuje vlastní model pro ukládání a přenos výukového obsahu. Mohou tak do výuky zahrnout své specifické požadavky, které jim SCORM použít neumožňuje a docílit vyšší integrity vlastních materiálů se systémem.

Další generací SCORM představuje **TIN Can API**, což je zcela nová specifikace pro vzdělávací technologie. Jejím cílem je zachycovat široké spektrum aktivit od uživatelů a využívat ho pro zefektivnění vzdělávacího procesu. [4]

2.3 Adaptivní vzdělávání

I když výuka u e-learningu probíhá individuálně, neohlíží se na studijní potřeby každého žáka. Výukové opory jsou sepsány jednou a v lineární návaznosti prezentovány studentovi. Student však danou problematiku ze skript nemusí pochopit způsobem, jakým je sepsána. V takovém případě se bohužel nemůže ihned obrátit na učitele nebo na spolužáka, aby mu to podrobně a názorně vysvětlil. K dispozici má většinou pouze email na učitele nebo diskuzní fórum.

Na tuto problematiku se váže studie zvaná Adaptivní vzdělávání, jejímž cílem je adaptovat výukový proces v e-learningu individuálním potřebám a znalostem studentů. Pro

účely automatické adaptace výuky, byly rozpoznány dva základní typy studentů: holisti a detailisti.

- **Holisté** uchopují učivo jako celek. Umí vytvářet a používat obecná pravidla a vidět učivo v širším měřítku.
- **Detailisté** postupují při učení po malých částech, které musí nejdříve plně pochopit, aby se dopracovali k obecnému pojetí učiva. Při učení postupují logicky a řídí se raději lokálními pravidly než těmi obecnými.

Někteří studenti mohou také více preferovat teoretický výklad a jiní zase raději praktickým odvozováním a experimentováním. Přístupy k učení se mohou u žáků lišit i z hlediska hloubkového stylu.

- **Hloubkový styl** učení – žák je vnitřně motivován a učivo se snaží pochopit do hloubky, později si ho stále dobře pamatuje
- **Povrchový styl –** žáka učení nebaví (vnější motivace) nebo na něj nemá moc času a proto volí nejrychlejší způsob zapamatování učiva nazpaměť, látku si pamatuje jen dočasně

Volba učebního stylu závisí na žákovi i na schopnostech pedagoga. Pokud učitel dokáže žáky dobře motivovat a vzbudit v nich zájem, pak studenti často volí hloubkový přístup. V opačném případě nebo v situaci, kdy žák pociťuje ohrožení a úzkost, pak volí povrchový přístup.

Toto hloubkové rozdělení se u adaptivní výuky používá zejména proto, aby žáci, kteří učivo ovládají lépe a mají rychlejší tempo vstřebávání informací, nemuseli procházet učivo stejně do detailu jako ti, kteří s ním mají problémy nebo o něj mají hlubší zájem.

Cílem adaptivního vzdělávání je vytvářet studijní opory zohledňující tyto základní typy studentů a díky automatizovaným procesům v e-learningových systémech ho pak prezentovat studentům v takové formě, která je pro ně nejvhodnější a vede k nejefektivnějším výsledkům. Tím jak docílit individuálního režimu u e-learningu se zabývá následující kapitola. [7]

2.3.1 Proces personalizace výuky u e-learningu

Aby výuka mohla probíhat individuálně je nutný sběr informací týkajících se studenta a úprava materiálů pro adaptivní účely.

Sběr osobních informací

Od studenta je nutno znát informace, jako jsou jeho znalosti v daném oboru, osobnostní typ a učební styl, který student preferuje. Tyto informace si systémy mohou získat prostřednictvím dotazníků a sledováním jeho aktivit při vykonávání kurzu. Charakteristiky studenta jsou následně uloženy do databáze a použity pro řízení procesu výuky.

Struktura materiálů pro adaptivní výuku

Důležité je upravit i vzdělávací materiály pro účel adaptivní výuky. Struktura těchto studijních materiálů vyžaduje, aby učivo bylo rozděleno na malé části (kapitoly, lekce, rámce), a aby bylo sepsáno v několika variantách odpovídajících různým učebním stylům. Díky rozčlenění materiálů na malé části je možno sestavit výuku podle individuálních požadavků.

Varianty učiva se budou lišit pouze zaobalením faktických informací. Studentovi je na základě jeho znalostí nabízena odpovídající varianta učiva v preferované hloubce. Například pokud student odpovídá vizuálnímu typu, bude mu nabídnuta verze učiva s vysokým obsahem obrázků a názorných ilustrací. V případě verbálního typu bude v učivu obsaženo více textu apod.

Všechny výukové části musí být opatřeny podrobnými metadaty, popisující o jaký typ učiva se jedná a pro koho je vhodné. Metadata jsou potřebné v systému pro dohledání a výběr správného učebního stylu podle zadaných charakteristik studenta.

Adaptivita v systémech

Adaptivní LM systémy by měly umožňovat vytvářet větvenou posloupnost materiálů. Linie průchodu adaptivním kurzem se může různě rozcházet a opět spojovat do jednoho bodu podle struktury přichystaných variant.

Pokud kurzy umožňují vytvářet libovolné struktury materiálů, je nutná implementace vnitřní logiky a podmínek, které budou řídit výukový proces. Vnitřní logika na základě přijatých informací o studentovi sestaví ideální výukový plán a určí nevhodnější způsob průchodu kurzem.

Systémy často obsahují tzv. rozcestníky, kde se cílenými otázkami zjistí typ studenta a na základě jeho odpovědi se zobrazí individuální výukový materiál. Průběh kurzem může být ovlivněn výsledkem z testu, přímou odpovědí nebo i odpozorovaným chováním. Sofistikované systémy mohou evidovat čas strávený nad řešením úloh a pročítáním výkladových části. Tyto hodnoty lze využít pro vylepšení učebního profilu studenta. [8]

Metodika adaptivity má však i své nevýhody, a tou je například časová náročnost tvorby studijních opor. Výukový materiál totiž musí být sepsán v mnoha verzích a logicky na sebe navazovat. Další nevýhodou je snížení lidské adaptivnosti. Pokud bude učivo dítěti podáváno pouze jedním způsobem, nebude schopno se v jiné variantě učiva vyznat a adaptovat se na ni. [9]

3 Stávající řešení

Na internetu a i na trhu jsou k dispozici stovky e-learningových nástrojů, které umožnují tvorbu vzdělávacích kurzů. K těm známějším patří například Moodle, který je představitelem LMS i LCMS a autorský nástroj Adobe Captivate.

3.1 Moodle

Moodle je open-source řešení LMS. Dnes je jeden z nejpoužívanějších systémů pro vzdělávání studentů na vysokých školách. V ČR je více jak 100 registrovaných serverů používajících Moodle. Tento systém splňuje standarty SCORM 1.2 a IMS pro sdílení obsahu a může také běžet na všech platformách (Windows, Linux, Mac,…).

Princip Moodle je takový, že každá škola má nainstalovanou svou instanci Moodlu, na které shromažďují studijní opory potřebné pro výuku. Některé školy používají Moodle primárně pro testování žáků, které díky automatickému vyhodnocení je velmi efektivní.

Systém Moodle 2.0 nabízí následující funkcionality:

- Zakládání výukových kurzů
- Vkládání digitálního studijního materiálu v různých formách (textová, multimediální, interaktivní).
- Tvorba testů s automatickým vyhodnocením výsledku: multiple choice, pravda / nepravda, esej, krátká odpověď, numerické úlohy a další typy úloh
- Možnost odevzdávat zadané úkoly na server
- Nastavení podmínek pro plnění činností - adaptivita
- Obousměrnou komunikaci mezi lektorem a žákem – pomocí diskusního fóra, vlastního emailu, chatu, anket apod
- Nastavování přístupových práv – nabízí několik uživatelských rolí (student, učitel, admin, host), kterým se zvlášť nastavují práva, například pro přístup do kurzů
- Evidence výsledků studentů a jejich hodnocení
- Možnost přizpůsobit vzhled a rozložení celého systému

3.1.1 Uspořádání kurzu

Kurz je v Moodlu základním stavebním prvkem výuky. Do jednotlivých kurzů, lišících se svým zaměřením, se ukládají všechny studijní materiály a činnosti pro studenty.

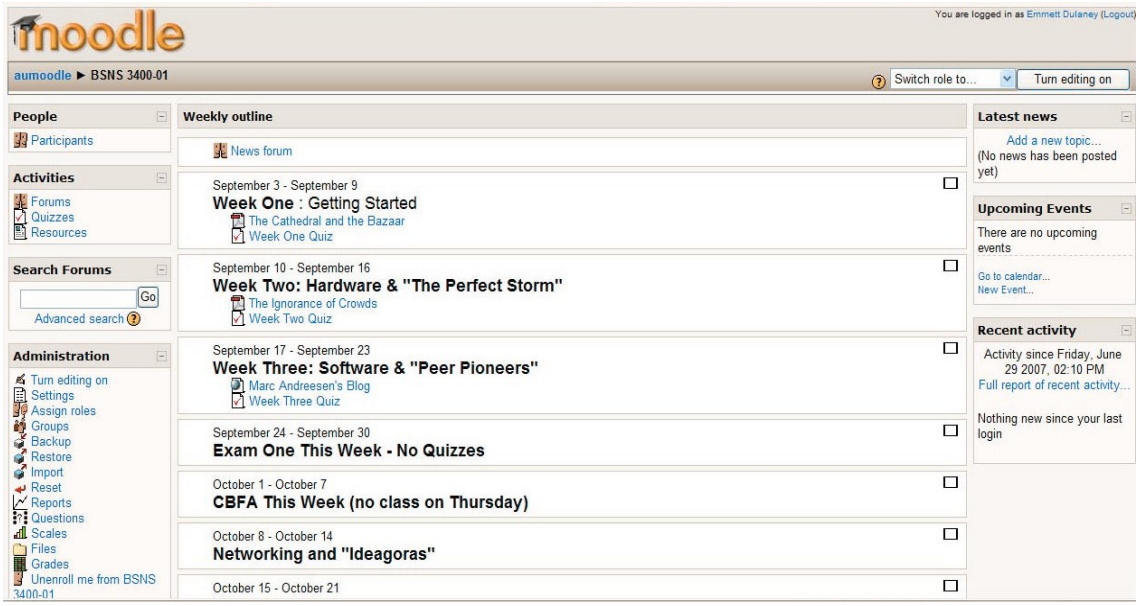
Studijní materiály odpovídají skriptům a jiným multimédiím uloženým na serveru za účelem předávání znalostí studentům v rámci výuky. Mohou mít formu textovou (PDF, Word dokument, HTML stránka, Power Point), multimediální (obrázek, video, animace) i interaktivní (aplikace ve Flashi, Javě, Javascriptu, apod.). Tvorba těchto materiálů probíhá mimo systém Moodle, pomocí rozhraní pro nahrávání souborů.

Č***innosti*** v kurzu umožnují studentům se aktivně podílet na výuce. Každá činnost má svůj specifický účel a slouží k vytvoření zpětné vazby mezi studenty a tutorem. K činnostem patří nástroje jako *chat*, *fórum, anketa*, *testy* a *úkoly*. Testy s automatickým vyhodnocováním slouží tutorům k snadnému vyzkoušení a oznámkovaní všech účastníků kurzu. Všechny činnosti odpovídají zásuvným modulům rozšiřujícím funkcionalitu systému Moodle.

Každý kurz je členěn na ***oddíly***, které slouží pro třízení výukového obsahu. Oddíly mohou mít například týdenní uspořádání, kde jsou probírané látky roztřízeny podle týdnů. [18]

3.1.2 Tvorba kurzů v prostředí Moodlu

Aby mohl uživatel vytvořit kurz, musí mít platný účet v dané instanci Moodlu a musí mít administrátorem přidělená oprávnění k tvorbě kurzů. Kurz se při vytváření zařazuje do odpovídající kategorie. Vyplní se krátký stručný název a popis. Následující rozhraní na obrázku 3.1 je zobrazeno uživateli, když se nachází uvnitř kurzu.



Obrázek 3.1: *Rozhraní systému Moodle*

Uprostřed obrazovky jsou zobrazeny všechny studijní materiály, testy a jiné činnosti strukturované do oddílů, které mají v tomto případě týdenní rozvržení (*Week One, Week Two, ...*). Oddíly pak spadají pod časový harmonogram a vytváří celou osnovu kurzu. V levém sloupci se nachází administrátorské nástroje pro správu kurzu, které jsou zobrazeny pouze uživatelům s autorskými přístupovými právy. V pravém menu mohou být umístěny různé okna pro zobrazování novinek, nadcházejících událostí apod. Vzhled Moodlu se dá přizpůsobit vlastním potřebám, ale jen do určité míry (změna stylů, přidání odebrání některých komponent, apod.)

Vytváření testu

Jednou ze základních funkcí systému je funkce testování. K tomu slouží aktivita typu Quiz. Při vytváření testu se přidává tato aktivita k jednomu z oddílu. Před samotnou tvorbou testu je nutno nejdříve vytvořit zvlášť otázky.

Otázky se ukládají se do tzv. *question bank*, které představuje globální úložiště pro všechny výukové materiály. Otázky mohou být strukturované do mnoha typů. K nejpoužívanějším patří například Multiple Choice, Short Answer, Pravda/Nepravda, Přiřazovací úloha, Esej a další.

Otázky jsou následně vybrány z banky a přiřazeny k testu. U testu se dále nastavují parametry jako časový limit na test a otázku, čas a datum, kdy se má test otevřít a zavřít, počet otázek na stránku nebo způsob vyhodnocení testu. Po vytvoření testu je jeho aktivita zahrnuta do oddílu spolu s ostatními materiály a čeká na jeho vykonání.

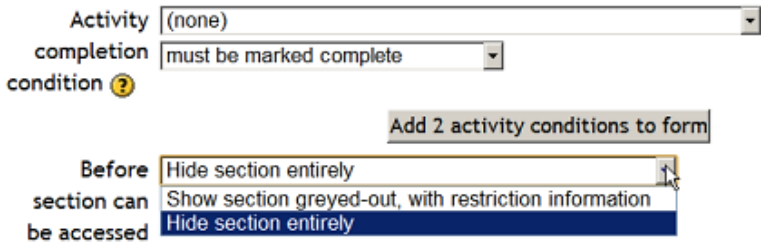
Podmíněné vykonávání aktivit

Moodle nabízí i pokročilé funkcionality, jako je například nastavení podmíněného přístupu k aktivitám. Podmínky definují případ, za jakého má být aktivita viditelná a přístupná. Omezení přístupu může být závislé například na datu, známce studenta, nebo na dokončení jiných aktivit.

Přístup k aktivitě může být omezen následujícími podmínkami:

- **Datetime condition** – určuje odkdy dokdy má být aktivita zpřístupněná
- **Grade condition** - určuje minimální a maximální hranici pro procentuální výsledek ze zvolené aktivity
- **Activity completion condition –** určuje závislost na vykonání předchozích aktivit
- **User field** – porovnává hodnotu ze zvoleného vstupního pole se zadaným řetězcem

Pokud chceme například, aby byl test vykonán pouze v případě, kdy je test z předchozího oddílu vykonán, vyplníme následující podmínku *Activity completion condition* zobrazenou na obrázku 3.2.



Obrázek 3.2: *Podmínka Activity completion*

Funkcionalita podmíněného přístupu je žádoucí hlavně při vytváření zmíněných adaptivních kurzů. Díky nim lze ovlivňovat průběh kurzem na základě hodnot získaných od účastníka kurzu. [10]

3.2 Adobe Captivate

Firma Adobe vyvinula pro účel e-learningu svůj vlastní desktopový nástroj zvaný Adobe Captivate. Tento nástroj byl vyvinut pro tvorbu vzdělávacích kurzů a screencastů, které jsou prezentovány v podobě HTML ve webovém prohlížeči.

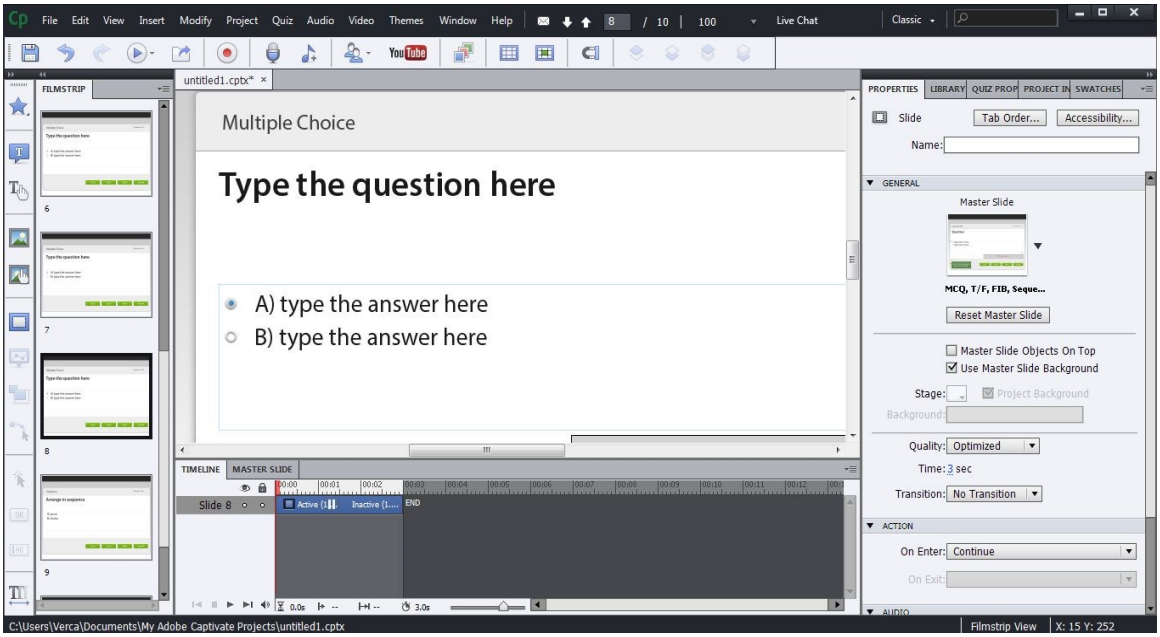
Adobe Captivate je velmi podobný nástroji Power Point od společnosti Microsoft, určeného pro vytváření prezentací. Jednotlivé obrazovky zvané slidy lze navíc obohatit o vzdělávací prvky jako kvízy, doplňovačky a další. Průběh prezentací může být v Adobe Captivate ovlivněn samotným uživatelem a jeho zpětnou vazbou.

Vzhledem k tomu, že struktura je velmi podobná prezentaci, lze výukový materiál vytvářet i přímým importováním Power Point prezentace ve formátu .PPT(X). Upravený výsledný kurz lze pomocí tohoto nástroje převést do výstupů typu .SWF (Flash) a do HTML 5. Díky formě HTML 5, tak může být obsah spuštěn na jakémkoliv mobilním zařízení ať už Android, IOS nebo Windows Phone. Má tudíž podporu všech mobilních platforem.

Adobe Captivate se řadí k systémům, které mají plnou podporu e-learningových standartů SCORM a AICC. Vytvořené výukové kurzy mohou být použity i v jiných vzdělávacích systémech mimo nástroj Adobe. [12]

3.2.1 Tvorba výukových materiálů v prostředí Adobe Captivate

Uživatelské prostředí je velmi podobné prostředí Power Point, jak můžete vidět na obrázku 3.3. Na levé straně jsou znázorněny jednotlivé slidy, které se budou žákovi zobrazovat v časové souslednosti. Na všechny lze aplikovat výchozí vzhled, který si uživatel může sám nastylovat, stejně jako způsob jejich prolínání.



Obrázek 3.3: *Uživatelské rozhraní Adobe Captivate 7*

Při vytváření nového slidu si uživatel vybírá z několika typů obrazovek. Každá z nich dokáže pojmout jiný typ obsahu. Typy slidů jsou následující:

- **New slide** - nový prázdný slide s výchozím vzhledem
- **New slide form** – nový slide s volbou rozmístění formulářových polí
- **Blank slide** – prázdný slide (bez aplikovaného výchozího tématu)
- **Question slide** – slide s otázkami, který se používá pro ověření znalostí žáků
- **Power point slide** – možnost vložení .ppt prezentace k ostatním slidům
- **Recording slide** – sleduje vaši aktivitu na obrazovce a nahrává ji do záznamu, který se při spuštění slidu přehraje
- **Image slide** – zobrazí zvolený obrázek v plné velikosti na obrazovce
- **Animation slide** – možnost vložení vlastní animace ve formátu .gif nebo .swf

S těmito typy lze vytvořit plnohodnotné kurzy. K dispozici je i panel nástrojů, který umožnuje na slidy vkládat obrázky, tvary, textová pole, vstupní pole a další komponenty.

Aby se tento nástroj odlišil od Power Pointu, obsahuje tzv. **question slidy**, které přidávají do prezentací interaktivitu a umožnují testování studentů. K výběru je zde několik možností testovacích slidů, k nimž se řadí všechny standardně používané úlohy, jako je multiple choice, short answer, přiřazovací úloha a mnoho dalších. Stejné typy úloh nalezneme i ve zmiňovaném nástroji Moodle. Speciálním typem je zde například otázka „hot spot“, která slouží k identifikaci správného místa na obrázku s nastavenou odchylkou.

Při vkládání vybraného question slidu do prezentace se zvolí počet otázek. Zadaný počet odpovídá počtu přidaných slidů s vybranou úlohou. Na konci vložených slidů s otázkami je automaticky vložen jeden vyhodnocovací slide, který zobrazuje konečný výsledek.

Pokud chceme otázky zobrazovat v náhodném pořadí použijeme **Question pool**, který představuje zásobník pro otázky. Question poolu můžeme vytvořit i více než jeden a lze je z jednoho projektu importovat do druhého. Když jsou všechny otázky vložené v poolu, pak stačí do prezentace vložit slide typu Random question a vybrat pool s vloženými otázkami.

Jednotlivé testovací slidy mají i své podrobnější nastavení, díky kterému lze otázky lépe nastylovat, zvolit způsob vyhodnocení testu a počet pokusů na odpověď. Pokročilejším nastavením je možnost zvolit si konkrétní akci, která nastane po odeslání odpovědi. Nastavená akce může být i podmíněna. Lze tak nastavit jednu akci, když uživatel odpoví správně a jinou akci, když odpoví špatně. Princip podmíněného vykonávání aktivit odvíjeného od znalostí žáka je základním stavebním prvkem pro adaptivní vzdělávání.

3.2.2 Typy akcí

Akce odpovídají různým činnostem, kterými lektor může řídit průběh kurzu. Níže je uveden jejich seznam.

- Přechod na jiný slide
  - ***Continue***
  - ***Go to the previous slide***
  - ***Go to next slide***
  - ***Go to slide last visited***
  - ***Jump to slide*** - skočí na vybraný slide z nabídky
- ***Open URL or file*** - otevře zadanou URL adresu nebo soubor
- ***Send e-mail to*** – pošle email na zadanou adresu
- ***Execute Javascript***
- ***Play/Stop Audio*** – přehrát a zastavit audio výstup
- ***Execute Advanced Action*** – pokročilé akce rozeberu níže
- Akce se aplikuje na vybranou komponentu (tlačítko, label, obrazec, ..)
  - ***Show*** – zobrazí
  - ***Hide*** - skryje
  - ***Enable*** – povolí
  - ***Disable*** – zakáže

- Akce týkající se nadefinovaných proměnných
    - ***Assign*** – přiřadit proměnné hodnotu
    - ***Increment*** – zvýšit proměnnou
    - ***Decrement*** – snížit proměnnou
- Akce ovládající průběh prezentace
    - ***Pause***
    - ***Exit***

Díky zmíněným akcím lze v prostředí Adobe Captivate vytvářet složitější struktury kurzu s více variantami průchodu. Pro specifičtější potřeby zde navíc slouží nabídka ***Advanced Action***, která s použitím vlastních proměnných umožnuje vytvářet podmínky, za kterých se má akce vykonat.

3.2.3 Použití proměnných a nastavení pokročilých akcí

Adobe Captivate je vhodným nástrojem i pro tvorbu složitějších kurzů, které nemají pouze lineární průchod, jak to bývá u prezentací.

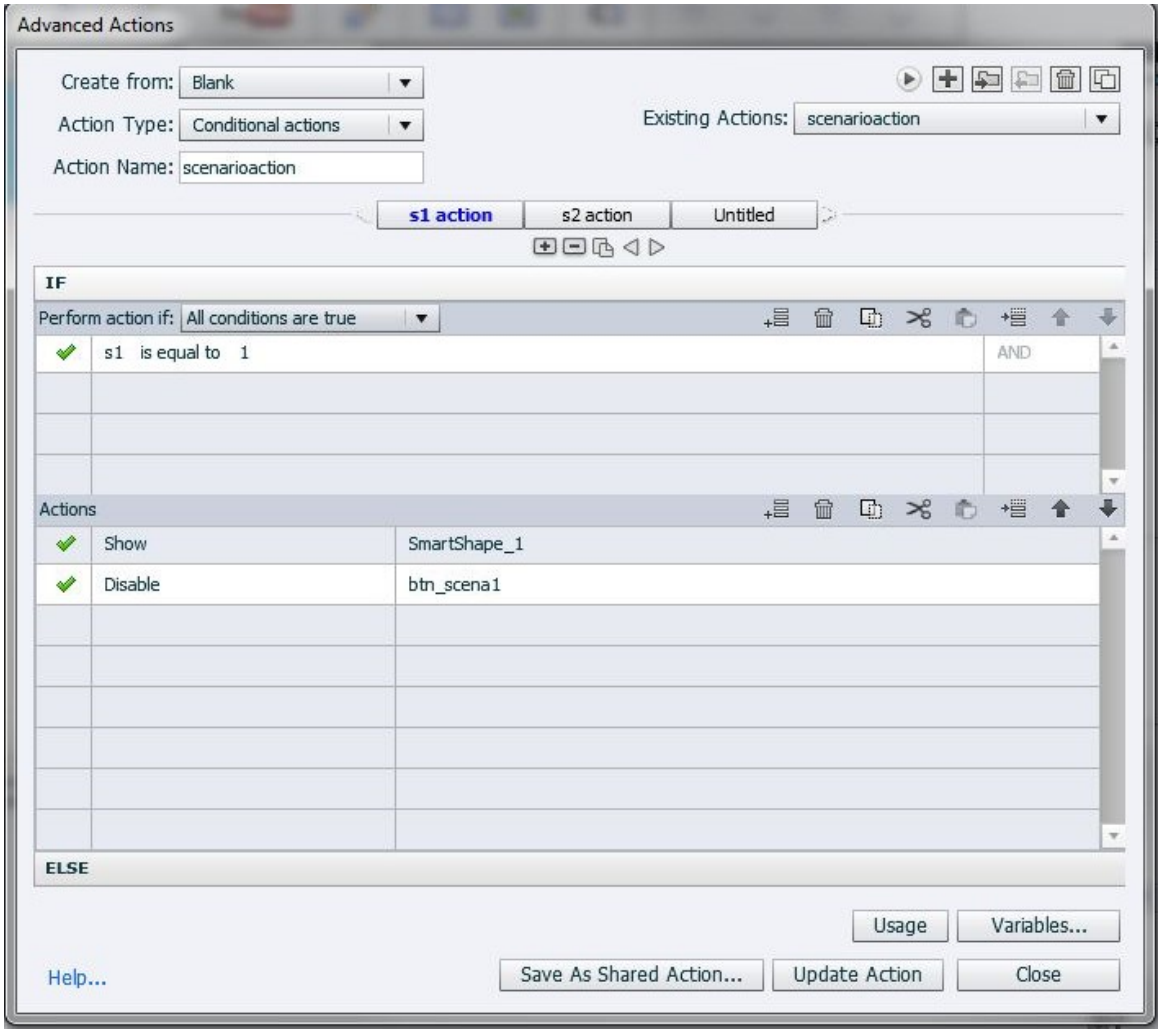
Pokud chceme, aby průchod kurzem byl různě rozvětvený nebo aby spuštění jednoho slidu bylo podmíněné vykonáním jiných, neobejdeme se bez vytváření vlastních proměnných a podmínek. Díky nim lze nastavit co, kdy a jak se má vykonat.

Variables (proměnné)

Variables zde slouží k uchování hodnot jako je hlavního skóre, ID uživatele, číslo aktuálního slidu, název prezentace a mnoha dalších výchozích hodnot, které se automaticky generují při vytváření prezentace. Můžeme si zde vytvořit i vlastní proměnné a jejich hodnotu v průběhu kurzu měnit.

Advanced Actions (pokročilé akce)

Advanced Action je formulář, který nám slouží k zadávání podmínek a akcí, které se mají při jejich vyhovení vykonat. Díky pokročilým akcím lze ovlivnit průběh kurzu na základě hodnot proměnných. Na následujícím obrázku 3.4 lze vidět, jak jeho rozhraní vypadá.



Obrázek 3.4: *Rozhraní Advanced Action*

Jednotlivé podmínky (akce) se ukládají zvlášť. Pro ukázku jsou zde vytvořeny dvě akce – „*s1 action*“ a „*s2 action*“. V první části se definují podmínky a v druhé části akce, které se mají při jejich splnění vykonat.

K tvorbě podmínek slouží tři vstupní pole, kde první a poslední obsahuje vždy nějaký literál nebo proměnnou a prostřední pole obsahuje operátory (>,<,=, !=, <=, >=) napsané slovně pro lepší představivost. Jednotlivé podmínky jsou pak pomocí logických operátorů AND (a zároveň) a OR (nebo) spojeny do jedné hlavní podmínky.

V druhé části při tvorbě akcí se vyplňují jen dvě pole a to akce, která se má vykonat (show, disable) a komponenta, na kterou se má akce aplikovat (tlačítko, obrazec, slide, …). Tímto principem mohou vznikat adaptivní kurzy v prostředí Adobe Captivate.

3.2.4 Odesílání výsledků

Kurzy vytvořené v nástroji Adobe Captivate jsou studentům nejčastěji prezentovány formou HTML. Student tudíž obdrží URL adresu s kurzem, který má vykonat. Po vykonání kurzu je student vyzván, aby vyplnil své identifikační údaje. Jeho výsledky jsou následně odeslány na centrální místo, které má pod správou lektor. Adobe Captivate nabízí možnost odesílat výsledky kurzu na několik umístění, například na vlastní interní server, Moodle, Acrobat.com, do jiných LMS nebo do Adobe Connect.

Každá volba vyžaduje nakonfigurování daného serveru a spojení. Pro zhlédnutí výsledků může posloužit i vlastní přidružená aplikace ***Adobe Captive Quiz Result Analyzer***, která stáhne aktuální výsledky a umožňuje je prohlížet a spravovat v uživatelsky přívětivé formě. [11]

Adobe Captivate lze používat pouze jako autorský nástroj pro tvorbu výukových kurzů v SCORM standartu. Spouštění kurzu a získávání výsledků lze zcela ponechat na jiném LMS, které dokáží lektorům nabídnout více funkcí pro řízení výuky.

3.3 Zhodnocení systémů

Adobe Captivate i systém Moodle jsou velmi sofistikované nástroje pro vytváření studijních opor. Na obou řešení spolupracují týmy profesionálních pracovníků z celého světa, na čemž se odráží i jejich vysoká kvalita. Oba nástroje jsou ale zcela odlišné – jak svou funkcionalitou, tak i provedením.

Moodle je určen zejména pro organizaci výuky, k testování žáků a k vytváření sdíleného výukového obsahu. Obsahuje všechny nástroje, které učitel potřebuje k tomu, aby mohl vést dálkové studium.

Adobe Captivate je autorský desktopový nástroj vyvinut pro účel vytvoření jednorázového kurzu a jeho následné distribuci. Nezabývá se tolik zpětnou vazbou účastníků, jako spíše formou a designem s jakým je kurz prezentován. Na rozdíl od kurzů tvořených v Moodlu, které mají pevně danou strukturu, lze v Captivate vytvořit kurz přesně podle svých představ.

Oba nástroje prezentují svůj výstup v HTML formě, která umožňuje snadnou přístupnost účastníkům z jakéhokoliv zařízení. Také podporují výukové standarty jako je SCORM, které zajišťují kompatibilitu s ostatními LMS.

3.4 Inspirace pro vlastní řešení

Zmíněné nástroje Moodle a Adobe Captivate a jejich způsob řešení byl využit při návrhu vlastního systému. Cílem práce není vytvořit zcela obdobný nástroj, ale využít některé přínosné vlastnosti obou systémů a skloubit je do jednoho originálního řešení.

První přínosnou vlastností, která by měla být zahrnuta ve výsledném řešení je univerzální charakter výukových struktur a kurzů. V tomto případě se bude systém odvíjet od Adobe Captivate. S využitím proměnných a nastavováním pokročilých akcí, tak bude moct uživatel vytvářet libovolné struktury s různým způsobem jejich procházení. Kurz ale nebude složen z jednotlivých slidů, ale z výukových komponent, které v sobě budou hierarchicky zanořeny.

Centrální úložiště výukových materiálů dostupné na internetu je druhým bodem, který bude řešen podle systému Moodle. Pro účely ukládání materiálů bude vytvořen datový model, stejně tak pro účely ukládání výsledků z testů. Uživatel bude do kurzů přidávat předem vytvořené výukové úlohy strukturované do skupin, které bude vybírat z tzv. banky otázek. Moodle je navržen způsobem, aby umožnoval správu uživatelů a nastavování přístupových práv ke kurzům. Tímto směrem by se mělo odvíjet i vlastní řešení.

Výhodou vlastního řešení oproti zmíněným systémům bude jeho otevřenost. Nástroj bude dostupný on-line v jediné instanci všem uživatelům internetu a bude nabízet globální přístup k vytvořeným kurzům i k autorskému nástroji.

4 Návrh a analýza systému

4.1 Specifikace zadání

Cílem této práce je navrhnout a naimplementovat nástroj pro vytváření vzdělávacích struktur s možností prezentovat vzdělávací obsah i v mobilních zařízeních. Řešení by mělo pokrývat problematiku jak různorodý multimediální obsah strukturovat a ukládat v rámci systému a jak zajistit jeho přístupnost i z jiných nezávislých umístění, např. z mobilního zařízení.

Výsledný vzdělávací systém se bude skládat ze dvou hlavních částí. První bude tvořit nástroj pro vytváření výukových struktur a druhý bude sloužit k prezentaci a spouštění struktur na různých zařízeních. Systém odpovídá spojení LMS a LCMS. Přístupnost všech výukových materiálů bude zajištěna použitím internetových technologií při implementaci nástroje.

Součástí práce je i navržení struktury doručovaného obsahu, která musí mít univerzální charakter, aby dokázala obsáhnout různé varianty vzdělávacího obsahu. Struktura výukového materiálu by měla být natolik univerzální, aby při přidání nové varianty úlohy nebylo nutno měnit strukturu databáze, do které se materiál ukládá. Stejně tak i nástroj pro zobrazování výukových materiálů by měl být lehce rozšiřitelný o nové typy úloh bez nutnosti přepisovat stávající kód.

Obsah materiálů bude rozdělen do kurzů, jejichž struktura bude hierarchicky rozdělena od těch největších nezávislých celků po ty nejmenší nazvané výukovými objekty. Výukové objekty musí dodržet zásadu znuvupoužitelnosti, která umožňuje zahrnout objekty do jiných kurzů bez nutnosti je znovu vytvářet.

Testovací část systému bude obohacena o zpětnovazební prvky, na jejichž základě bude koordinován průchod skrz kurzem. Procházení výukových objektů v rámci kurzu nemusí být pouze lineární, ale může se různě větvit a adaptovat na schopnosti uživatele. Tato schopnost adaptivního vzdělávání bude také částečně zahrnuta ve výsledném řešení.

4.2 Funkční požadavky

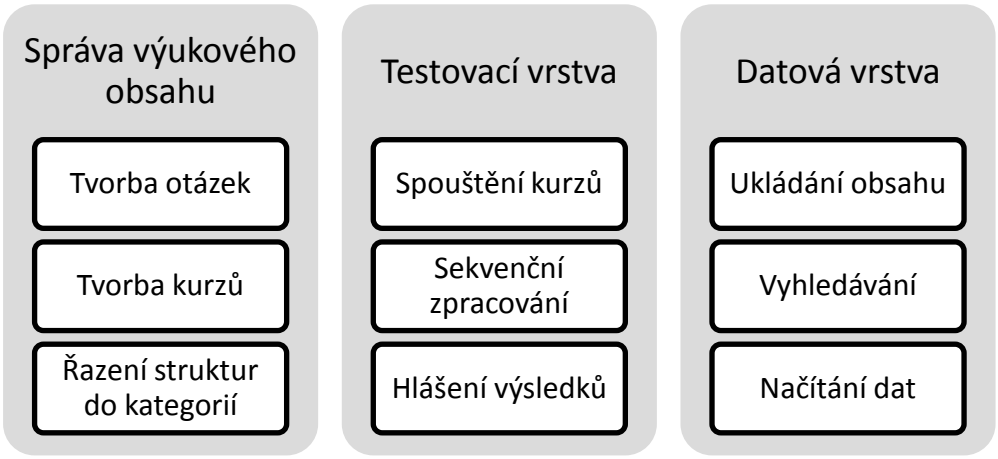
Nástroj by měl splňovat následující požadavky, které se zčásti odvíjí ze e-learningových zásad SCORM standartu popsaných v kapitole 2.2.1.

- Vytváření různorodých výukových struktur
- Znovupoužitelnost výukových komponent i v jiných kurzech
- Přístupnost a nalezitelnost výukových materiálů na internetu
- Způsob procházení kurzem podporující adaptivitu
- Spouštění vytvořených výukových kurzů

- Možnost testování studentů s použitím zpětnovazebních prvků
- Automatické vyhodnocování výsledků studenta
- Přizpůsobení prezentační vrstvy pro běh na mobilních zařízeních

4.3 Členění systému

Navržený systém se dělí do tří hlavních vrstev zobrazených na následujícím schématu.



Obrázek 4.1: *Schéma výukového systému*

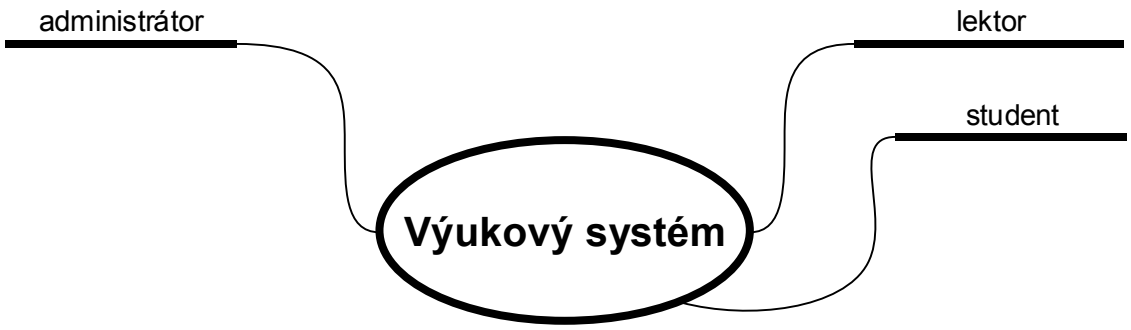
První část systému tvoří obslužný prvek určený k vytváření a správě multimediálního obsahu. Umožňuje vytvářet různorodé výukové struktury v podobě výkladů a interaktivních testů propojených do celistvých kurzů. Jednou z priorit zadání bylo, aby výsledný systém byl co nejuniverzálnější a umožňoval vytvořit velkou škálu vzdělávacího obsahu s různými způsoby jejich procházení. Pro snadnější dohledatelnost vytvořených struktur se obsah řadí do kategorií.

Druhou částí je testovací vrstva, jejímž hlavním účelem je zobrazovat vytvořené výukové struktury v časové souslednosti a získávat od uživatele zpětnou vazbu. Vrstva je navržena pro spouštění kurzů a automatické vyhodnocení odpovědí. Sekvenční zpracování kurzu může být ovlivněné znalostmi a chováním uživatele. Testovací vrstva proto obsahuje logiku pro jednoduché adaptivní vzdělávání.

Poslední částí, která je nedílnou součástí předchozích vrstev, je datová vrstva. Jedná se o databázi a její obslužnou vrstvu, která má na starost ukládání a načítání výukového obsahu, výsledků a jiných potřebných dat. Datový model je podrobně popsán v kapitole 4.6.

4.4 Uživatelé systému

Se systémem bude pracovat několik typů uživatelů, z čehož vyplývají následující tři role zobrazené v kontextovém diagramu.

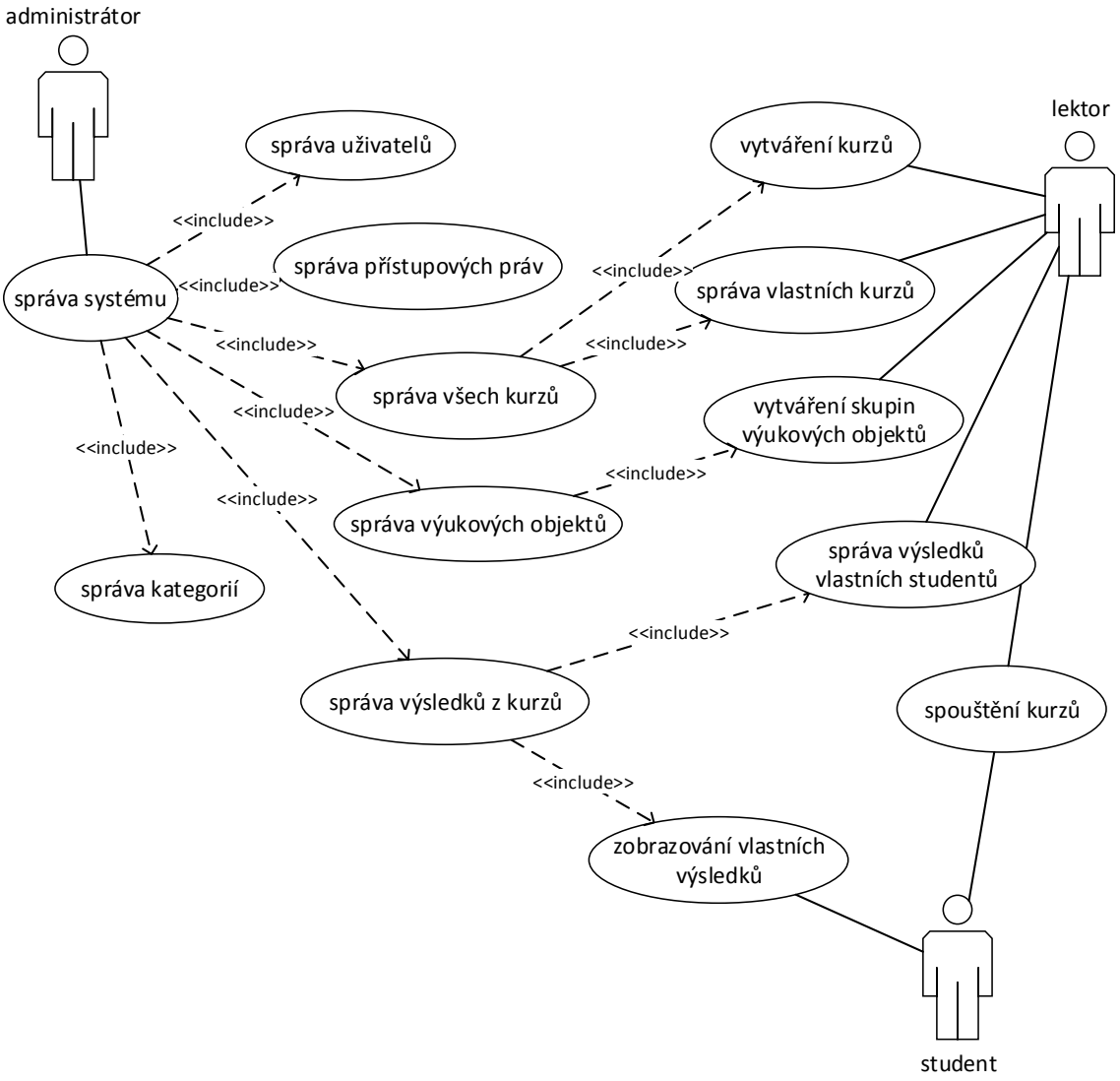


Obrázek 4.2: *Kontextový diagram uživatelů systému*

Každá role má své vlastní přístupové práva k jednotlivým částem systému.

- **Administrátor** – má nejvyšší práva a tudíž neomezený přístup ke všem funkcím systému, hlavním úkolem administrátora je spravovat výukové materiály a jejich kategorie, uživatele systému a další entity
- **Lektor** – práva mu umožňují vytvářet a spravovat vlastní výukové objekty a zpřístupňovat je svým studentům
- **Student** – přistupuje k prezentační vrstvě systému a využívá všechny funkce s tím spojené

Následující Use Case diagram ukazuje, k jakým funkcím systému má každý uživatel přístup.



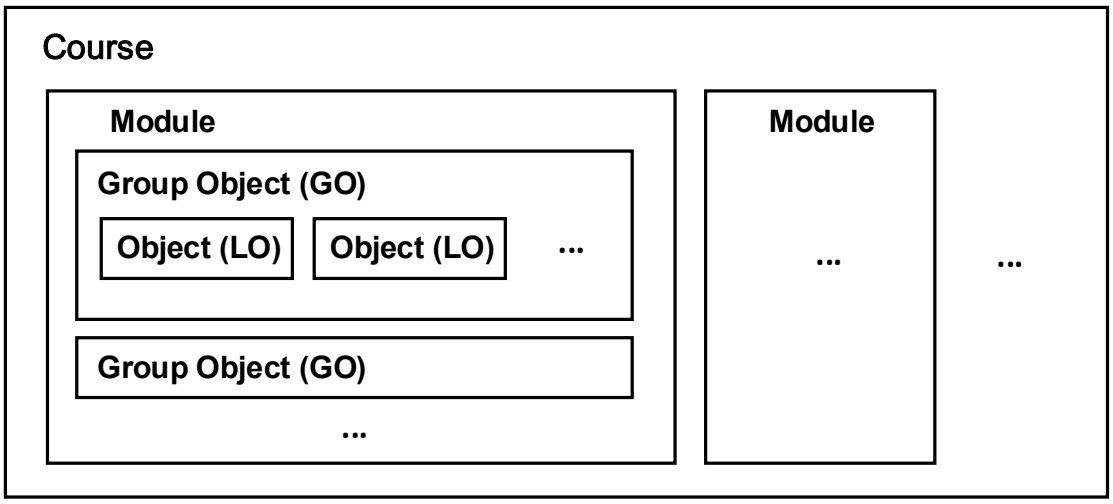
Obrázek 4.3: *Use Case diagram*

4.5 Návrh struktury doručovaného obsahu

Pro ukládání výukového obsahu byl navržen model skládající se z několika typů komponent, které jsou v sobě hierarchicky zanořeny. Rozčleněním kurzu na jednotlivé dílčí části je možno dosáhnout větší univerzálnosti a také částečné adaptivity systému. Při návrhu struktury bylo dodrženo pravidlo znovupoužitelnosti, aby bylo možno vytvořené výukové materiály použít i v jiných kurzech a souvislostech.

Kurz se skládá z následujících výukových komponent:

- **Výukový objekt – *Learning Object (LO)*** – nejmenší komponenta, která obsahuje jednotlivé multimédia (texty, obrázky, animace) strukturované do různých typů úloh.
- **Skupina objektů** – ***Group Object (GO)*** – seskupuje výukové objekty do stejnorodých logických celků odpovídajících probírané tématice
- **Modul – *Module*** – je univerzální výuková komponenta určená pro definování struktury kurzu a návazností mezi skupinami otázek. Modul je základním stavebním prvek v kurzu a díky vnitřní logice s ním lze dosáhnout určité adaptivity.
- **Kurz – *Course* –** nejvyšší typ komponenty, která představuje samotný kurz a zaobaluje všechny předchozí komponenty do jedné. Obsahuje základní informace o názvu a obsahu kurzu.



Obrázek 4.4: *Uspořádání kurzu*

Na obrázku 4.4 lze vidět hierarchické uspořádání komponent. Výukové objekty (LO) spolu se skupinami objektů (GO) tvoří základ pro uchování výukových materiálů a tvoří se zvlášť. Obsahují jednotlivé otázky a odpovědi, výkladové části, multimédia a další.

Při tvorbě kurzu se definují moduly, kterým se mohou přiřazovat předem vytvořené skupiny otázek. Díky modulům a jejich vazbám se určuje celá struktura kurzu. Kurz všechny komponenty zaobaluje do jedné celistvé a obsahuje obecné informace týkající se jeho obsahu a nastavení.

4.5.1 Výukový objekt (LO)

Výukové objekty (LO) jsou dále nedělitelné výukové části, které zaobalují jednotlivé multimédia (texty, obrázky, animace) do struktur příkladů. Příklady mohou mít jakoukoliv podobu, např. textové otázky a otevřené odpovědi, uzavřené otázky s variantami odpovědí (multiple-choice), doplňovací nebo přiřazovací úlohy a mnoho dalších. Pro splnění základních potřeb byly do systému zahrnuty tři typy následujících úloh označeny zkratkou pro snadnou identifikaci.

- **Výklad - „v“** – úloha obsahující výkladové texty a materiály které nejsou určeny pro testování ale pouze pro čtení
- **Otevřená otázka - „t“** – testovací úloha obsahující otázku a pole pro otevřenou textovou odpověď
- **Multiple choice - „m“** – testovací úloha s jednou otázkou a více variantami odpovědí

Všechny typy úloh je nutno uchovat v jedné struktuře – entitě, kterou je možno uložit do databáze. Pro tyto účely je nevhodnější značkovací jazyk XML, který umožnuje uchovat data v jednotném formátu. Každý typ úlohy má tak svou vlastní XML strukturu, lišící se svým uspořádáním.

XML struktura pro uchování výkladu:

```
<task type="v">
  <innerHTML> /* text otázky v HTML jazyce */  </innerHTML>
</task>
```

XML struktura pro uchování otevřené otázky:

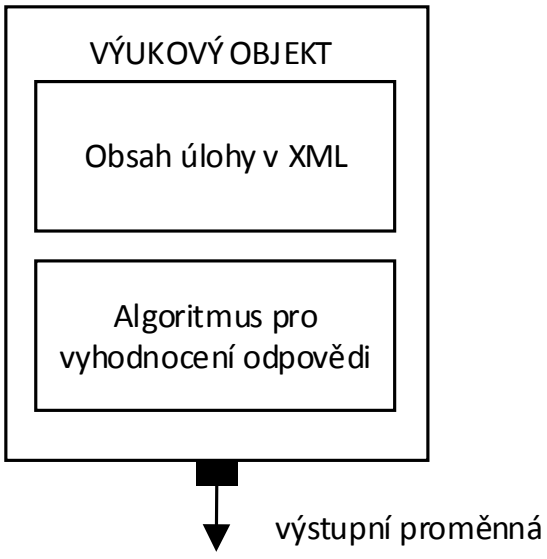
```
<task type="t">
    <question> /* text otázky v HTML jazyce */  </question>
    <answer> /* text odpovědi */ </answer>
</task>
```

XML struktura pro uchování multiple choice:

```
<task type="m">
   <question>  /* text otázky v HTML jazyce */ </question>
   <answers>
      <answer correct="t">  /* text odpovědi */ </answer>
      <answer correct="f">  /* text odpovědi */ </answer>
      <answer correct="f">  /* text odpovědi */ </answer>
   </answers>
</task>
```

Texty úloh mohou být i různě naformátované. Formátování je důležité hlavně v případě, že se jedná o dlouhé souvislé texty, které by jinak byly špatně čitelné. Pro účely formátování textu je použit jazyk HTML s tagy pro font, styl, barvu a velikost písma.

Každý typ výukového objektu má mimo XML definovaný i algoritmus pro vyhodnocení správnosti odpovědi. Pokud se jedná například o úlohu typu „t“ bude algoritmus srovnávat získanou odpověď uživatele se zadanou správnou odpovědí. Výstupní proměnná z vyhodnocení této LO bude true nebo false. U více variant odpovědí, může být výsledek po provedení výpočetního algoritmu v procentuální formě a výstup bude tedy odpovídat proměnné typu double. Výstupní proměnné generované výukovým objektem jsou použity při finálním vyhodnocení modulu. Na obrázku je znázorněné schéma výukového objektu.



Obrázek 4.5: *Schéma výukového objektu*

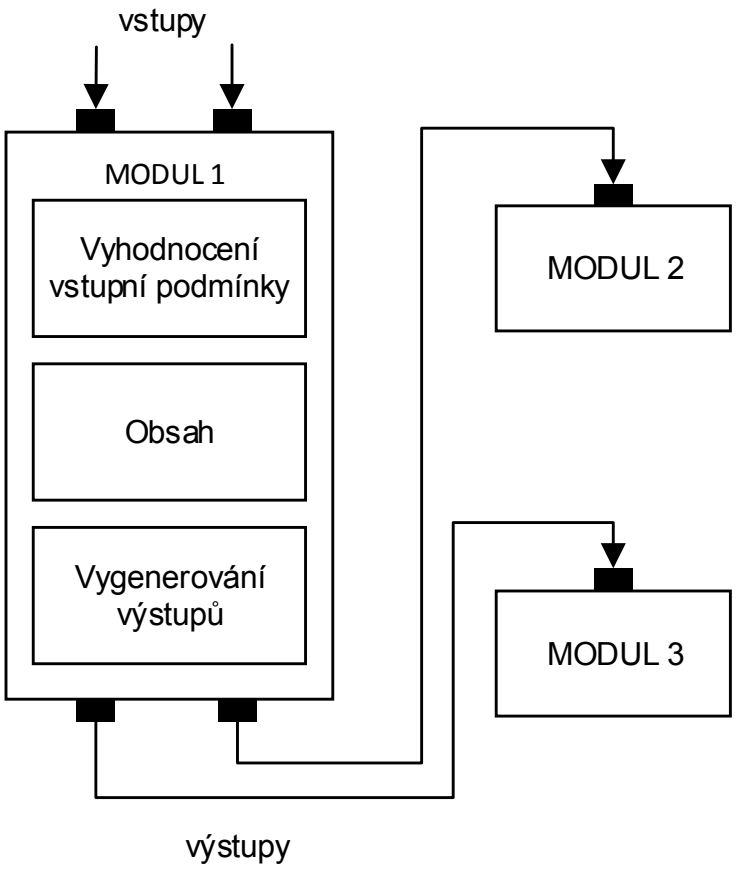
4.5.2 Modul

Modul představuje univerzální komponentu, definovanou svými vstupními a výstupními parametry. Výukový modul byl navržen z důvodu, aby rozčlenil kurz na samostatné výukové části, mezi nimiž je možno definovat vazby, a tím určit strukturu celého kurzu.

Obsah modulu není pevně daný. Modul může sloužit k zobrazování již vytvořených výukových objektů nebo k výpočtu a zobrazování výsledků. Cílem bylo vytvořit tuto komponentu co nejuniverzálnější se společným rozhraním pro definování vstupních a výstupních hodnot.

Vstupy modulu odpovídají výstupním hodnotám jiných výukových modulů. Všechny moduly jsou vzájemně provázány předávanými proměnnými, které mohou představovat například výsledky z testů. Proměnné jsou uchovány ve spojnicích, kterými se určují vazby mezi moduly.

Na vstupní proměnné modulu může být aplikována vstupní podmínka, při jejímž splnění je možno modul vykonat. Schéma modulu vypadá následovně:



Obrázek 4.6: *Schéma modulu*

Vstupní a výstupní proměnné modulů

Vstupní a výstupní proměnné jsou přenášeny vazbami mezi moduly a mohou nabývat těchto hodnot:

- Procentuální nebo bodové vyjádření výsledku modulu
- Boolean hodnota zda-li uživatel modul vykonal/nevykonal (prošel až do konce)
- Uživatelské proměnné - číselné nebo textové proměnné zadané přímo uživatelem

Vyhodnocení vstupní podmínky

Když dojde ke spuštění kurzu, automaticky se zkontrolují vstupní podmínky u všech modulů. Vykoná se ten, jehož podmínky nebyly dosud splněny, ale nyní jsou. Pokud je těchto modulů více, dá se přednost tomu s nastavenou vyšší prioritou.

Vstupní podmínka je složena ze výstupních proměnných jiných modulů a logických operátorů AND a OR. Může mít například tento tvar:

- *Definice proměnných:*

```
Double Input1 = Module-1.result;
Double Input2 = Module-2.result;
```

- *Definice algoritmu:*

```
(Input1  > 50)  AND  (Input2 < 20)
```

Modul se stejnou podmínkou se vykoná v případě, kdy výsledek z prvního modulu je větší jak 50 a zároveň byl druhý modul vykonán na méně jak 20 procent.

Obsah modulu

Při návrhu výsledného řešení byly vytvořeny dva typy modulů, jejímž cílem je splnit nejzákladnější potřeby uživatelů při vytváření kurzu. Tyto typy modulů se liší svým obsahem a účelem. Patří sem:

- **Testovací modul** – seskupuje a zobrazuje skupiny objektů
- **Výsledkový modul** – vypočítává finální výsledky z přijatých vstupních hodnot a zobrazuje je na výsledkové listině

Moduly mohou být v budoucnu rozšířeny o další typy, a tím nabízet uživateli větší škálu funkcionalit.

Generování výstupů

Při vykonání obsahu modulu a jeho dokončení dochází k vyhodnocení a doplnění jeho výstupních hodnot. Toto dosazení může vést ke spuštění následujícího modulu, který tuto výstupní proměnnou očekával na svém vstupu. Dochází tak k průchodu kurzem, jehož linie se může různě větvit a adaptovat na výsledky uživatele.

4.5.3 Kurz

Kurz je nejvyšší spustitelnou komponentou v systému. Má na starost řízení kurzu od inicializace, přes provádění výukových modulů, až po její ukončení. Obsahuje všechny základní informace o kurzu a o jejím obsahu.

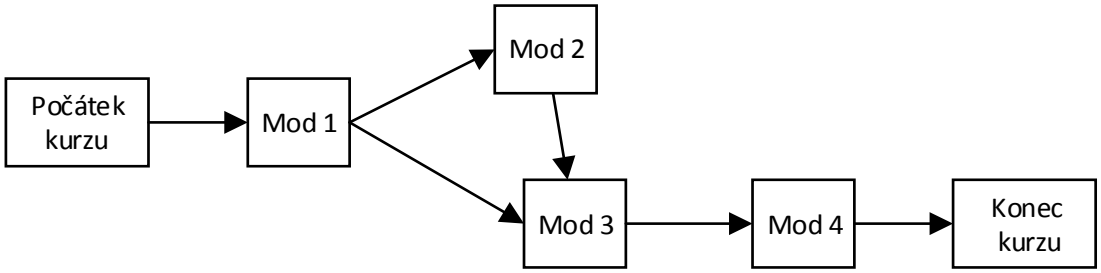
V kurzu se nachází dva výchozí speciální moduly, počáteční a ukončovací modul. Pokud má být zahájení kurzu omezené nějakými vstupními podmínkami, například datem a časem odkdy je kurz zpřístupněn, jsou tyto podmínky definované uvnitř počátečního modulu. To samé platí i pro výstupní podmínky kurzu. Výstupní podmínkou může být například úspěšné vykonání všech modulů uvnitř kurzu. V tomto případě bude tato podmínka obsažena na vstupu ukončovacího modulu. Pokud nelze spustit ukončovací modul, nelze dokončit ani kurz.

Kurz nakonec generuje jednu výstupní proměnnou – výsledek kurzu, která se váže k danému uživateli, který kurz vykonal.

Uspořádání kurzu

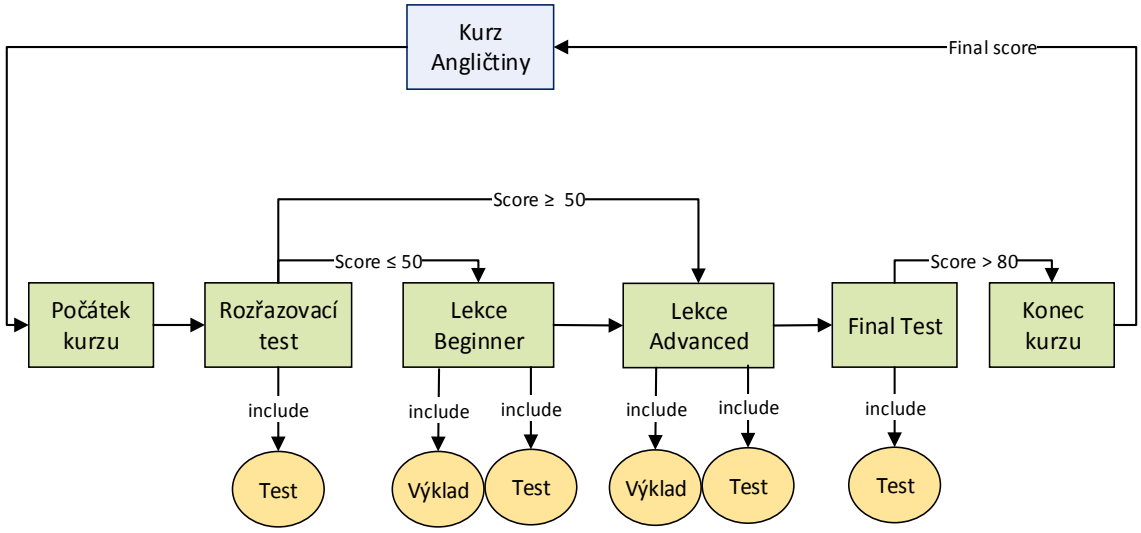
Uspořádání jednotlivých částí kurzu (modulů) nemusí být pouze lineární. Linie průchodu kurzem se může různě rozvětvovat a opět spojovat do jednoho bodu. Zvláště pokud se jedná o adaptivní výuku s připravenými variantami materiálů.

Schéma posloupnosti modulů v kurzu může odpovídat následujícímu obrázku.



Obrázek 4.7: *Schéma posloupnosti modulů v rámci kurzu*

Z obrázku předchozího schématu lze rozpoznat, že struktura kurzu je v podstatě vyjádřena orientovaným grafem. Uzly grafu tvoří jednotlivé moduly a hrany jsou tvořeny spojnicemi modulů s přenášenou výstupní proměnou. Jedinou výjimku tvoří smyčky, které se v tomto případě nemohou v kurzu vyskytovat. Spojnice modulu totiž nemůže vést na stejný modul, ze kterého vycházela.



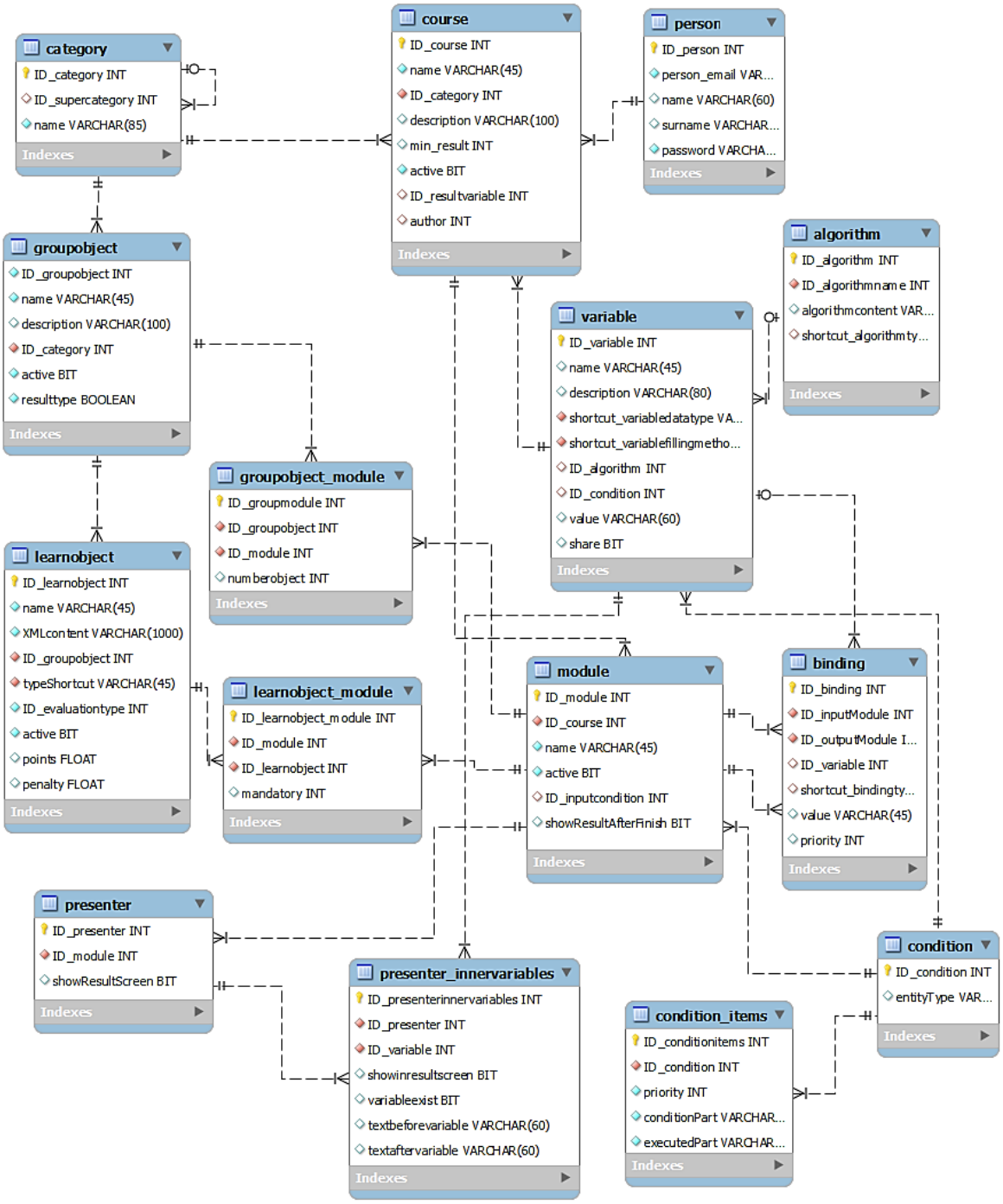
Obrázek 4.8: *Příklad uspořádání komponent v kurzu4*

Na obrázku 4.8 lze vidět, jak je kurz strukturován s jednotlivými komponentami.

- **Jednotka kurzu** je označena modře – v našem případě kurz Angličtiny. Kurz je úspěšně dokončen v případě, že z finálního testu získáme výsledek vyšší jak 80 bodů.
- **Výukové moduly** značené zeleně - představují jednotlivé lekce angličtiny. Provádí se v závislosti na vykonání jejich vstupních podmínek. Z rozřazovacího testu se můžeme dostat přímo do lekce Advanced, pokud měl uživatel z testu výsledek větší než 50 bodů. V opačném případě musí student nejdříve vykonat lekci Beginner a pak až může přejít na lekci Advanced.
- Žluté jsou **výukové objekty**, které zastupují strukturované úlohy. Tyto objekty jsou nezávislé na ostatních a mohou se vyskytovat v jakémkoliv modulu a kurzu. Například pro finální test mohou být použity náhodně vybrané otázky z testů předchozích.

4.6 Datový model

Struktura databáze se odvíjí od navržených komponent. Jednotlivé výukové objekty a moduly představují hlavní entity v databázi. Výřez struktury používané pro ukládání kurzů je znázorněn na následujícím schématu.



Obrázek 4.9: *Databázové schéma pro ukládání kurzů*

Třídy **Course, Module, GroupObject** a **LearnObject** byly popsány již v předchozích kapitolách.

Třída **Binding** – tvoří spojnici mezi moduly, obsahuje odkaz na vstupní a výstupní modul a také uchovává ID přenášené výstupní proměnné.

Třída **Variables** – reprezentuje samotné proměnné představující vstupy a výstupy modulů. Jsou součástí nejen modulů ale i algoritmů a vazeb. Může nabývat mnoha typů hodnot v závislosti na tom, jakou hodnotu uchovává. Způsob jakým se má výsledná hodnota vypočíst je dán atributem shortcut_fillingmethod, který dává k dispozici tyto možnosti:

- *Algorithm* – hodnota proměnné se vypočte pomocí algoritmu, jehož ID je definované uvnitř proměnné
- *Condition* – hodnota proměnné se vypočte na základě definované podmínky, jejíž ID je definované uvnitř proměnné
- *Value* – hodnota se rovná zadanému řetězci

Třída **Algorithm** – obsahuje konkrétní algoritmus pro výpočet proměnných. Obsah algoritmu je zapsán matematickou formulací s využitím proměnných.

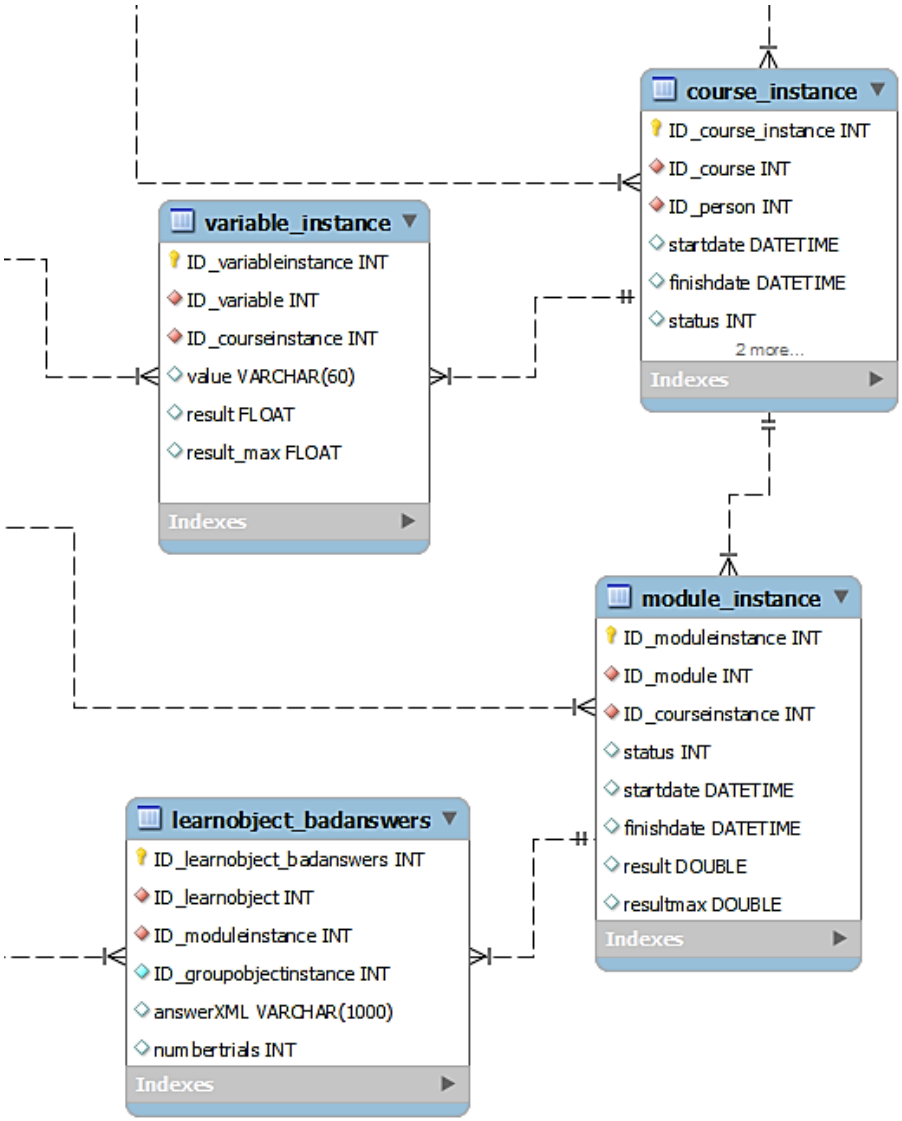
Třída **Condition** a **Condition_Items** - definuje algoritmus pro podmínku (conditionPart) a algoritmus pro vyplnění proměnné při splnění podmínky (executedPart).

Třída **Presenter** - odpovídá speciálnímu výsledkovému typu modulu

Třída **GroupObject_Module -** definuje obsažené skupiny objektů pro výukový typ modulu.

Třída **Person** – obsahuje informace o uživatelích systémů

Pro účel ukládání výsledků uživatele byly navrženy další entity, jejichž strukturu můžete vidět na následujícím obrázku. Jedná se o třídy **Course_instance, Module_instance, Variable_instance, Learnobject_badanswers.** S entitami se pracuje v prezentační vrstvě kurzu, ve které dochází k testování. Ukládají všechny hodnoty vznikající při provádění kurzu daným uživatelem v daném čase. K těmto hodnotám patří například status modulu, výsledky z úloh, kolik času strávil uživatel nad jakou úlohou apod. Tyto informace mohou později sloužit k analýze uživatelových znalostí a k určení studijního typu.



Obrázek 4.10: *Databázové schéma pro ukládání výsledků*

Obrázek kompletního databázového schématu se všemi použitými entitami je v příloze.

5 Implementace systému

Realizovaný systém má podobu webové aplikace dostupné on-line. Pro implementaci systému byla zvolena technologie ASP.NET, která je vhodná pro tvorbu složitějších webových aplikací. Systém je postaven na frameworku ASP.NET Web Forms a psán v jazyce C#. Některé funkcionality systému byly vyřešeny použitím existujících javascriptových knihoven, které umožnily snadný vývoj bez nutnosti řešit problematiku vykreslování grafů a jiných.

Řešení systému je nezávislé na platformě a je optimalizované i pro chytré mobilní telefony díky responzivnímu designu a s využitím CSS3 stylů.

5.1 Proces tvorby výukových materiálů

Struktura výsledného systému se skládá z již zmíněných navržených komponent modulů, skupin objektů a výukových objektů. V rámci tvorby kurzu je nutno tyto komponenty naplnit výukovými materiály a definovat mezi nimi návaznosti. To je úkolem lektorů, kterým je pro tyto účely nabízeno navržené uživatelské rozhraní.

Pro snadnou dohledatelnost výukových materiálů jsou všechny výukové komponenty v rámci kurzu přiřazené do kategorií, které tematicky odpovídají obsahu probírané látky. Kategorie zastupují předměty v klasické výuce a mohou se do sebe různě hierarchicky zanořovat.

Procesu tvorby výukových opor se skládá ze dvou hlavních fází. V první fázi musí lektor naplnit výukové objekty vlastním materiálem, který chce studentům prezentovat. V druhé fázi vytváří kurz a určuje jeho obsah a strukturu.

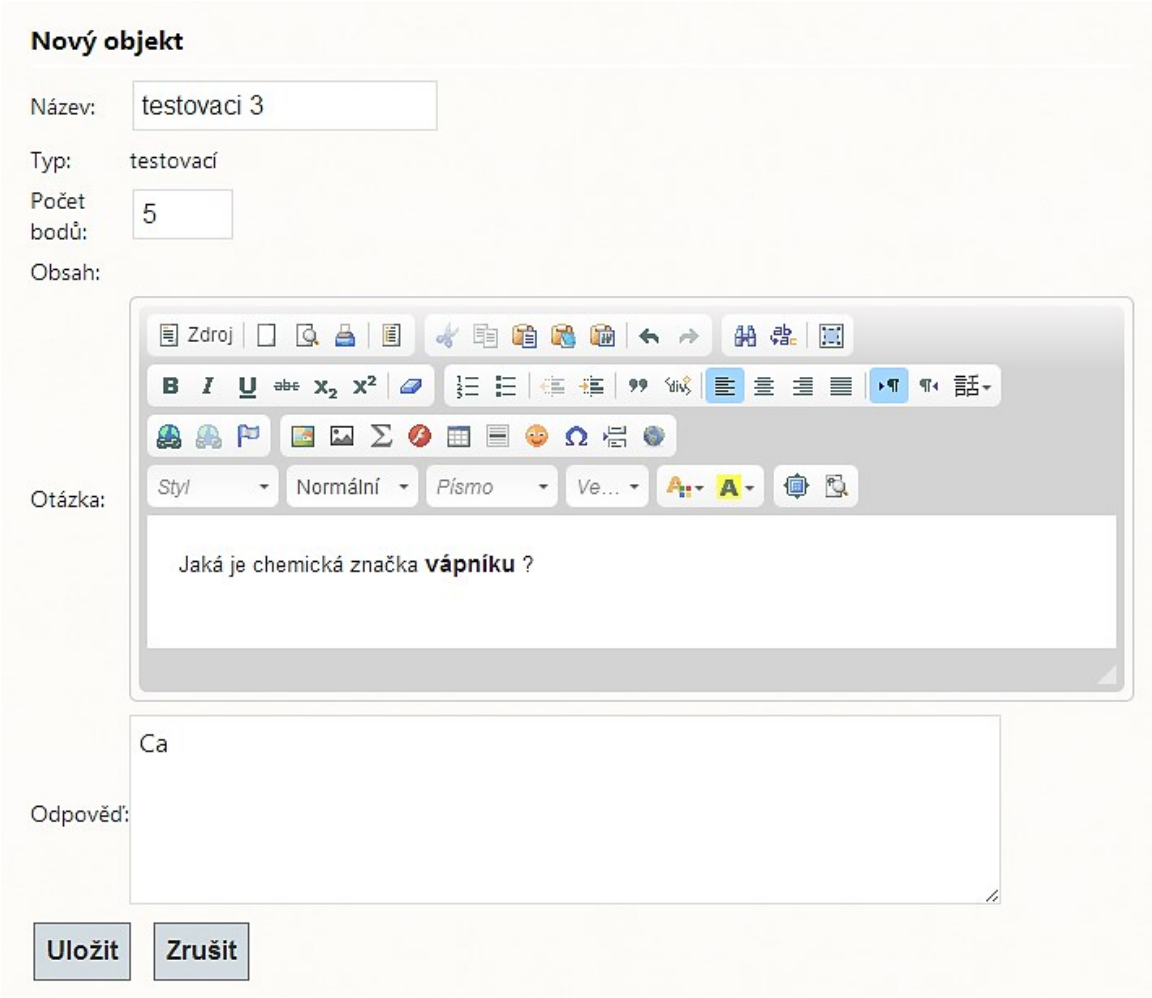
5.1.1 Vytváření skupin výukových objektů

Výukových objektů (LO) se ve výuce vyskytuje mnoho, a proto lektor vytváří celé skupiny objektů odpovídajících komponentě **group object** (GO).

Při vytváření nové skupiny objektů vybírá uživatel nejdříve kategorii, do které chce objekty zařadit. Poté určí název a popis skupiny a může do ní začít vkládat jednotlivé výukové objekty. K dispozici jsou zatím dva typy objektů:

- **Výkladový** – obsahuje nastylovaný výkladový text, nedochází zde k žádnému vyhodnocení
- **Testovací** – obsahuje textovou otázku a pole pro otevřenou odpověď, při vyhodnocení se porovnávají dva textové řetězce a výsledkem je jedna pravdivostní hodnota true/false

Tyto typy objektů lze snadno rozšířit o další typy, s přidáním vyhodnocovací logiky dovnitř kódu systému a s metodami pro tvorbu a čtení XML struktury zaobalující nový typ úlohy. Rozhraní pro ukládání výukových objektů je zobrazeno na následujícím obrázku 5.1.



Obrázek 5.1: *Uživatelské rozhraní při vytváření výukového objektu*

U každého výukového objektu je definován jeho název a počet bodů, které uživatel získá při správné odpovědi. Obsah objektu se liší podle zvoleného typu. Na znázorněném obrázku 5.1 je zvolen testovací typ úlohy. Lektor vyplňuje textovou otázku a správnou odpověď, kterou na otázku očekává. Otázka by měla být co nejjednoznačnější, aby na ní bylo možno odpovědět jenom jedním způsobem.

Pro potřeby naformátování textů otázek a výkladů zde slouží jednoduchý textový editor zvaný *CKeditor*. Jedná se o open source knihovnu psanou v Javascriptu. Výstup editoru lze získat v HTML formě, kterou lze snadno ukládat a modifikovat.

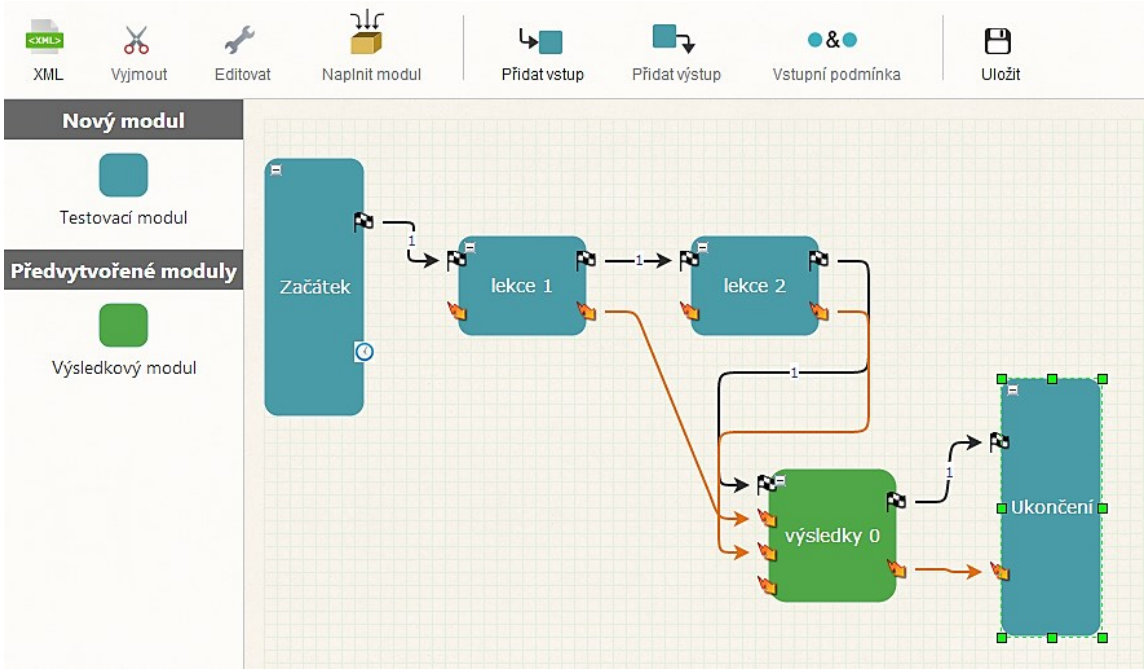
Každá úloha je při uložení konvertována do XML struktury a uložena do databáze. Po vložení všech výukových objektů dochází k druhé fázi, kterou představuje vytváření samotného kurzu. Jedná se o zcela oddělenou aktivitu. Výukové objekty mohou existovat samostatně bez kurzu, ale nelze je spustit. Tvoří pouze zdrojový výukový materiál, který je určen pro vložení do kurzu.

5.1.2 Vytváření kurzu

Vytváření kurzu může následovat, až když jsou předem připravené všechny výukové objekty. Při vytváření kurzu se vyplní základní informace týkající se jeho názvu popisu a zařazení do kategorie.

Lektor si následně zvolí skupiny, které chce do kurzu zahrnout. U vybraných skupin lze nastavit počet otázek (LO), které mají být ze skupiny náhodně vybrány. Pokud je tento počet menší než maximální může uživatel zvolit u každé otázky povinnost – jestli je povinná, nepovinná nebo jestli má být v kurzu vynechána.

V následujícím kroku jsou automaticky ze skupin otázek vygenerovány **moduly**. Moduly slouží k určení struktury kurzu. Pomocí editoru zobrazeného na obrázku 5.2 lze spojovat jednotlivé moduly orientovanými šipkami, a tím určovat linii průchodu skrz kurzem.



Obrázek 5.2: *Uživatelské rozhraní při vytváření struktury kurzy*

Jednotlivé **spojnice** mezi moduly přenáší hodnoty proměnných, jejichž hodnota může spustit průběh dalšího modulu. K hodnotám těchto proměnných se řadí například výsledky ze skupin otázek, časové údaje, hodnoty uživatele a další. Spojnice se vytváří přetahováním portů umístěných na krajích modulů. Porty stejně jako spojnice odpovídají třem základním typům:

- **Statusový typ** – značený vlajkou - určuje hlavní linii průchodu kurzem. Spojnice vedoucí ze statusových portů mají černou barvu a určují posloupnost, v jaké mají být moduly prezentovány uživateli. Při dokončení jednoho modulu je spuštěn modul následující, který je s aktuálním spojen statusovou vazbou. Pro docílení větvené posloupnosti kurzu je možno se spojnicemi odkazovat na víc než jeden modul. V tomto případě lze využít nastavování priority

statusových vazeb (číslo umístěné na spojnici). Vazba s prioritou číslo 1 je vykonána jako první, ty s vyšším číslem až poté. Pokud z jednoho modulu vede více vazeb se stejnou prioritou, může si uživatel vybrat jeden z následujících modulů sám.

- **Výsledkový** – značený bleskem – přenáší výsledky z testů a jiné hodnoty proměnných, které jsou vyžadovány následujícími moduly. Pokud má modul ve své vstupní podmínce odkaz na jednu z proměnných získaných od uživatele (např. výsledek předchozího modulu musí být větší než 50), pak musí být zmíněná proměnná i na vstupu modulu. Jedná se o tzv. lokální přístup k proměnným.

Při vytváření kurzu uživatel pracuje s moduly. Některé jsou automaticky vygenerovány s vloženými skupinami, další si ale uživatel může vytvořit sám. Nové moduly se přetahují z levého sloupce na plochu editoru. K dispozici jsou dva typy modulů: testovací modul a výukový modul.

Testovací modul

Testovací modul je typ modulu, který seskupuje skupiny objektů (GO) v jeden smysluplný celek, který si můžeme představit například jako lekci nebo rámec. Učivo by mělo být rozděleno do tak velkých celků, od kterých se očekává, že je student vyplní najednou, a od nichž chceme sledovat výsledky a pracovat s nimi i v rámci celého kurzu. Přesně taková část učiva by měla být obsažena i v testovacím modulu.

Při vykonávání tohoto typu modulu dochází ke spouštění jednotlivých příkladů (LO) z vybraných skupin objektů a mezi uživatelem a aplikaci dochází k interaktivní komunikaci. Pomocí úloh systém zjišťuje uživatelovy znalosti a schopnosti a jeho výsledky zaznamenává do databáze.

Když uživatel projde a vyplní všechny úlohy, dochází k závěrečnému vyhodnocení modulu a z výsledků všech LO je vypočtena jedna finální hodnota odpovídající výstupu modulu (výsledkový port).

Každý testovací modul může obsahovat *vstupní podmínky*, při jejichž splnění je modul spuštěn. Spuštění modulu může být omezeno například podmínkou, kdy uživatel musí dosáhnout určité hranice bodů z předchozí skupiny. Podmínky se píšou za sebe s využitím vstupních proměnných (hodnot z příchozích spojnic) a logických operátorů AND a OR.

Výsledkový modul

Výsledkový modul může sloužit k zobrazování tzv. *výsledkové listiny*. Jeho úkolem je na základě přijatých vstupních hodnot generovat nové výstupní hodnoty, které mohou odpovídat finálním výsledkům. Uživatel si v rámci výsledkového modulu může vytvořit libovolný počet nových proměnných a má na výběr, jestli chce jejich hodnotu zobrazit v listině na obrazovce a jestli vůbec chce zobrazit výsledkovou listinu.

Uživatel tento modul může používat pouze k vlastním dílčím výpočtům nebo k definování vlastních proměnných, které si nepřeje vůbec zobrazovat. Definování nové proměnné probíhá tak, že uživatel vyplní název proměnné a vybere z následujících možností, čemu proměnná odpovídá:

- Odpovídá jedné vnitřní proměnné
- Vypočte se pomocí defaultního algoritmu – za použití vstupních proměnných
- Algoritmus pro výpočet zadá uživatel sám – za použití vstupních proměnných
- Odpovídá textovému řetězci – uživatel přímo vyplňuje hodnotu proměnné

Takto definovaná proměnná může být navíc podmíněna vykonáním určité podmínky. V takovém případě se jedná o podmíněné zadávání proměnné a algoritmus pro její výpočet může vypadat následovně…

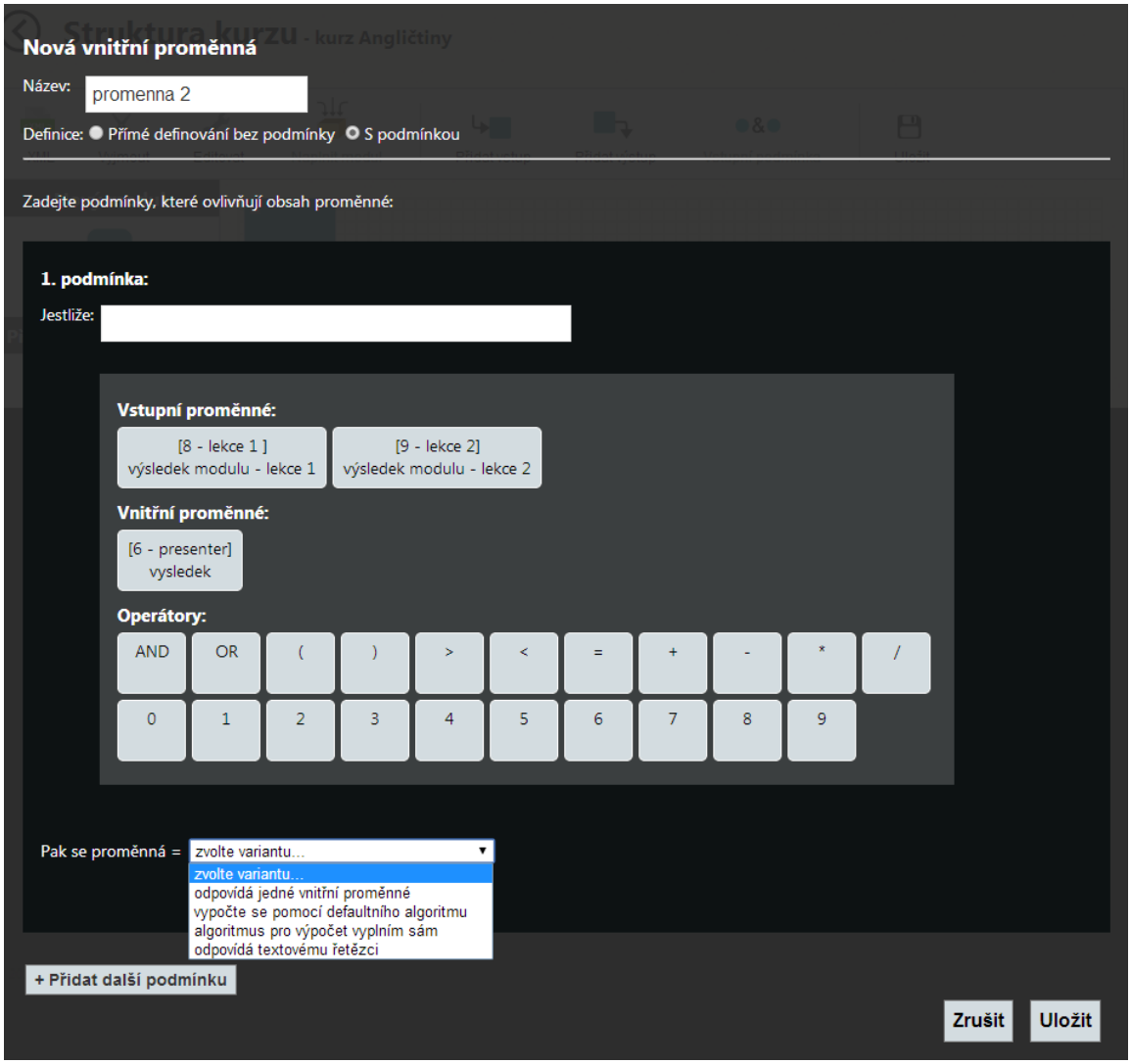
- *Definice proměnných:*

```
String in1  = Module-1.result;
Double output;
```

- *Definice algoritmu:*

```
if(in1 < 50){
    output = "nedostatečný";
}
else if (in1 >= 50 && in1 < 70)
{
    output = "dobrý";
}
else if (in1 >= 70)
{
    output = "výborný";
}
```

Rozhraní pro definici podmínek v systému je zobrazeno na následujícím obrázku. Při definování vstupních podmínek u testovacího modulu, neobsahuje podmínka druhou část *(„pak se proměnná rovná =“)*, protože proměnná je defaultně naplněna hodnotou true.



Obrázek 5.3: *Podmíněná definice proměnných ve výsledkovém modulu*

Podmínek je možno v systému definovat libovolný počet. Lze jimi zajistit například, aby byl finální výsledek kategorizován a vyjádřen slovně. Stačí určit několik rozmezí, při jakém se nová proměnná bude rovnat vyplněnému textovému řetězci a proměnou zobrazit ve výsledkové listině (viz. předchozí příklad).

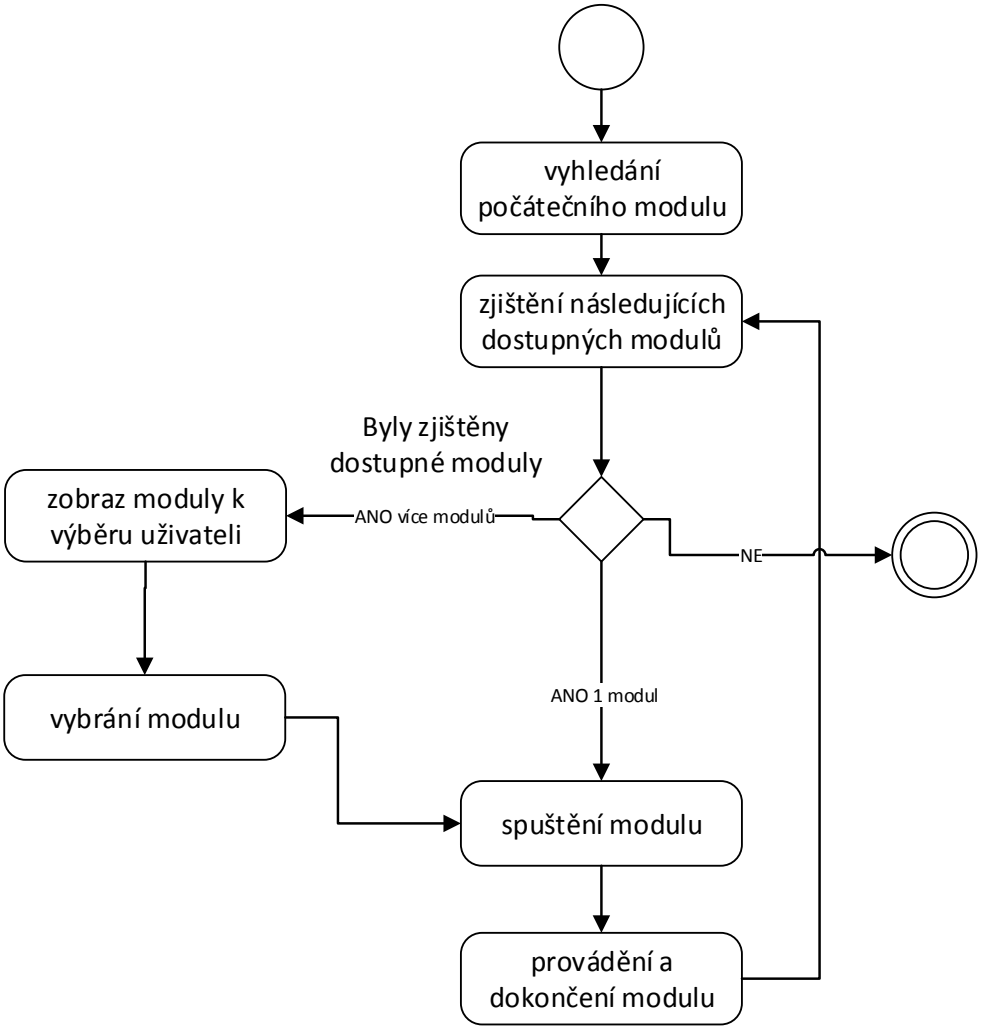
Výstupy výsledkového modulu tvoří nově nadefinované vnitřní proměnné, které se posílají dalším modulům na vstupy. Definování modulů, určování jejich vazeb a předávání proměnných jsou principy vedoucí k navržení celkové struktury kurzu. Další kapitola se zabývá spouštěním a vykonáváním kurzu.

5.1.3 Spouštění a vykonávání kurzu

Funkce spouštění a vykonávání kurzu náleží druhé vrstvě systému tzv. testovací/ prezentační vrstvě. K ní má přístup i samotný student. Uživatelské rozhraní systému nabízí studentovi seznam vytvořených kurzů, které může spustit a vykonat.

V detailu kurzu lze vidět informace jako je jeho název, detailní popis, jméno autora a seznam obsažených modulů v kurzu.

Při jeho spuštění dochází k inicializaci mnohých proměnných a vytváří se nová instance kurzu pro ukládání výsledků a odpovědí uživatele. Algoritmus pro vykonávání kurzu je zobrazen v následujícím diagramu aktivit.



Obrázek 5.4: *Diagram aktivit pro algoritmus vykonávání kurzu*

Při provádění kurzu je nejdříve zjištěn počáteční modul (označen v databázi atributem start). U počátečního modulu jsou zjištěny jeho odchozí statusové vazby a moduly, které se na jejich koncích nachází. Funkce zjišťování následujících modulů je docela komplikovaným algoritmem obsahujícím mnoho dalších aktivit včetně vykonávání vstupních podmínek, jejichž

problematice se věnuje další kapitola. Výsledkem funkce je vždy seznam modulů, které mohou být uživatelem spuštěny. Pokud je těchto modulů více je výběr konkrétního modulu ponechán na uživateli. Vybraný modul je spuštěn a výukový materiál obsažený v modulu je předkládán uživateli v náhodné posloupnosti, dokud uživatel neprojde obsah dokonce. Poté jsou výstupní proměnné modulu doplněny získaným výsledkem a proces se opakuje téměř od začátku. K ukončení kurzu dochází v případě, že byly všechny moduly vykonány a žádný následující modul už neexistuje.

Využití reflexe při vyhodnocení vstupních podmínek

Vstupní podmínky pro vykonávání modulů jsou zadané pomocí vnitřních proměnných kurzu a jsou zkombinované s logickými a matematickými operátory. Uložená struktura podmínky má v databází následující tvar:

```
_v_r_201 >= _v_r_202 || _v_r_201 > 80
```

Zápis proměnných s pomlčkami je přizpůsoben syntaxi jazyka C#. Pomocí reflexe je možné dynamicky za běhu aplikace sestrojit zdrojový kód a spustit ho. Tím lze podmínku vykonat a získat požadovaný výstup, který bude mít *true* nebo *false* hodnotu. Problematiku vykonání podmínky jsem chtěla řešit i pomocí speciálního matematického parseru *MathParserNet*. Nakonec se ale ukázalo, že knihovna nepodporuje logické operátory AND a OR, a proto jsem zvolila princip reflexe. Zvolený postup je navíc mnohem univerzálnější, protože umožnuje vykonat jakýkoliv sestrojený kód a může být použit i při podmíněné inicializaci proměnných u výsledkových modulů.

Proměnné obsažené v podmínkách a jiných algoritmech mají v systému předepsaný tvar, díky němuž lze konkrétní proměnnou snadno identifikovat a zjistit její hodnotu z databáze. Tvar se skládá ze dvou symbolů oddělených podtržítkem a číslem na konci.

- První symbol zastupuje název entity (v = variable, m = modul)
- Druhý symbol o jaký atribut se jedná (r = result)
- Třetí část složená z čísla odpovídá ID entity uložené v databázi

Proměnné se pomocí regulérních jazyků mohou rozparsovat na jednotlivé části a z databáze lze pak zjistit hodnotu konkrétní proměnné. Tvar regulérního výrazu je následující:

```
Regex rgx = new Regex(@"_\D_\D_\d*");
```

Celý algoritmus se i s třídou sestaví do jednoho textového řetězce, který vypadá následovně:

```
string formatstring = "namespace TrainMemory                    \n"+
                "{{                                             \n"+
                "   public class ExecuterCode                   \n"+
                "   {{                                          \n"+
                "        bool result = false;                   \n"+
                "        {0}                                    \n"+
                "        public bool executeCondition()         \n"+
                "        {{                                     \n"+
                "            {1}                                \n"+
                "            return result;                     \n"+
                "        }}                                     \n"+
                "    }}                                         \n"+
                "}} ";

string finalcode = String.Format(formatstring, initAllvariables,
initAllConditon);
```

Třída *ExecuterCode* inicializuje na začátku své proměnné. Výchozí proměnou je pravdivostní hodnota result nastavená na *false*. Do místa označeného {0} jsou vloženy inicializované proměnné, jejichž příklad je uveden níže.

```
double _v_r_201 =  58; _v_r_202 = 80;
```

Do místa označného {1} je vložena podmínka načtena z databáze v jejím původním tvaru ukázaném na začátku kapitoly.

Pro vykonání kódu uloženého v textovém řetězci bude využit mechanismus reflexe v prostředí ASP.NET, který umožnuje manipulaci a procházení objektového modelu konkrétní aplikace. Za běhu programu tak lze zjišťovat, jaké třídy jsou v aplikaci obsaženy, jaké mají metody a parametry, a přistupovat k jejich metadatům. Reflexe umožnuje s těmito strukturami manipulovat a dynamicky vytvářet nové třídy, což je vlastnost, které využijeme v následujícím příkladě použitém v projektu.

```
Dictionary <string, string> providerOptions = new Dictionary<string,
string>
{
      {"CompilerVersion", "v3.5"}
};
CSharpCodeProvider provider = new
CSharpCodeProvider(providerOptions);

CompilerParameters compilerParams = new CompilerParameters
{
        GenerateInMemory = true,
        GenerateExecutable = false
};
CompilerResults results = provider.CompileAssemblyFromSource
(compilerParams, finalcode);

object o = results.CompiledAssembly.CreateInstance
("TrainMemory.ExecuterCode");
MethodInfo mi = o.GetType().GetMethod("ExecuteCondition");

var result = mi.Invoke(o, null);
```

Textový řetězec *finalcode* je za běhu programu dynamicky sestaven a interpretován jako zdrojový kód, ke kterému lze přistupovat a spouštět jeho metody. Pomocí metod *getType*(), *getMethod*() a *Invoke*() lze v prostředí ASP.NET vyvolat požadovanou metodu *executeCondition*() a získat jeho výstupní proměnnou. Vyvolaná metoda je dynamicky přeložena, zkompilována a provedena přímo za běhu aplikace, což odpovídá principu reflexe.

Celý tento algoritmus byl použit při vyhodnocení vstupních podmínek a obdobně bylo řešeno i provádění algoritmu při výpočtu hodnoty proměnných.

5.2 Použité technologie

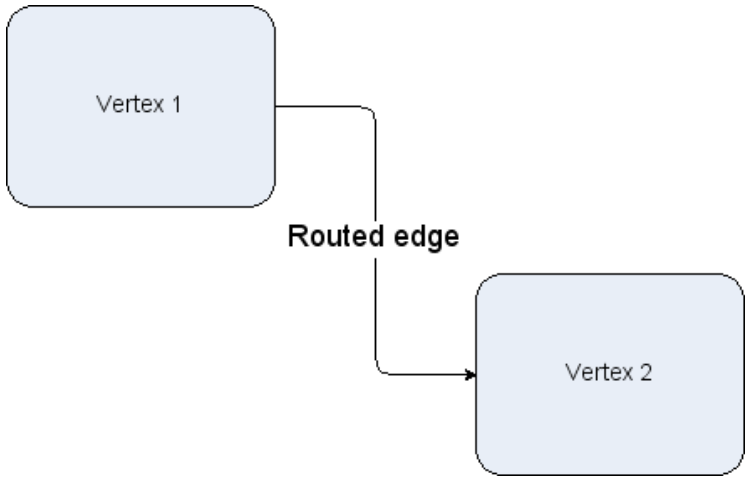
Systém je postaven na architektuře client-server. Realizace byla provedena s použitím frameworku ASP.NET Web Forms v kombinaci s jazykem C#. Aplikace založené na technologii ASP.NET jsou předem předkompilované a běží mnohem rychleji, na rozdíl od čistě skriptovacích jazyků.

Skriptovací jazyky Javascript a jQuery byly také zakomponovány do výsledného řešení. Při implementaci byla použita javascriptová knihovna *mxGraph* při navrhování struktury kurzu. Pro textový editor zobrazený při vytváření otázek byla použita jQuery knihovna *CKeditor*.

5.2.1 Knihovna mxGraph

Pro účely zobrazení modulů v designéru a přetahování spojnic mezi nimi byla použita knihovna ***mxGraph***. Knihovna je psaná v JavaScriptu a je určena pro vizualizaci diagramů. Je navržena pro webové aplikace, do kterých lze jednoduše zakomponovat. Běží na straně klienta v nativním prohlížeči a její výstup tvoří HTML 5.

Je vhodná pro tvorbu UML diagramů, digramů případů užití, workflow diagramů, databázových schémat, orientovaných grafů, schémat zapojení a mnoha dalších. Vizualizace grafu je založena na matematické teorii grafů, kdy se graf skládá z jednotlivých uzlů a vazeb. Ve zdejší terminologii jsou uzly nazvány **Vertex** a vazby **Edge.** Vazby jsou zde orientované a mají v sobě uložen odkaz na zdrojovou a cílovou buňku.



Obrázek 5.5: *Jednoduchý graf tvořený v knihovně mxGraph*

Ukázka kódu pro vytvoření grafu znázorněného na obrázku.

```
graph.getModel().beginUpdate();
try
{
  var v1 = graph.addVertex(parent,null,'Vertex 1',20,20,80,30);
  var v2 = graph.addVertex(parent,null,'Vertex 2',200,150,80, 30);
  var e1 = graph.addEdge(parent, null, 'Routed Edge', v1, v2);
}
finally
{
  graph.getModel().endUpdate();
}
```

Je zde několik typů interakcí, jakým můžeme schéma grafu upravovat, a to:

- Přetahováním buněk – měnění jejich pozice nebo přetahování z externího zdroje na plochu
- Klonováním buněk
- Měněním jejich velikosti a tvaru
- Editováním textů v buňkách

Knihovna tedy nabízí většinu potřebných funkcí, které jsou nutné pro vytváření a úpravu diagramů. Jednotlivé buňky lze nastylovat použitím třídy *mxStyleSheet*, která má v sobě namapovanou hash tabulků všech dostupných stylů.

Knihovna ke své funkci vykreslování diagramů používá vektorový grafický jazyk. Diagramy jsou pomocí knihovny převedeny do vektorového formátu, který se liší podle prohlížeče. SVG formát je standardem pro většinu prohlížečů až na Internet Explorer, v jehož případě je používán VML formát. Rozdílům všech běžně používaných prohlížečů je knihovna přizpůsobena a vývojářům nabízí jednotné API.

5.3 Přizpůsobení systému pro mobilní zařízení

Jedním z hlavních cílů práce je umožnit vzdělávání i na mobilních zařízeních. Lze tím dosáhnout větší užitné hodnoty systému, protože uživatel může provádět kurz kdykoliv a odkudkoliv, pouze s použitím svého mobilního telefonu.

Definice mobilního zařízení zahrnuje všechny malé přenosné bezdrátové přístroje, ke kterým se řadí např.: mobilní telefony, notebooky, netbooky, čtečky knih, PDA, tablety a další. Přizpůsobení systému má největší význam zejména pro chytré mobilní telefony (smartphony) a tablety, jejichž zastoupení na trhu neustále roste.

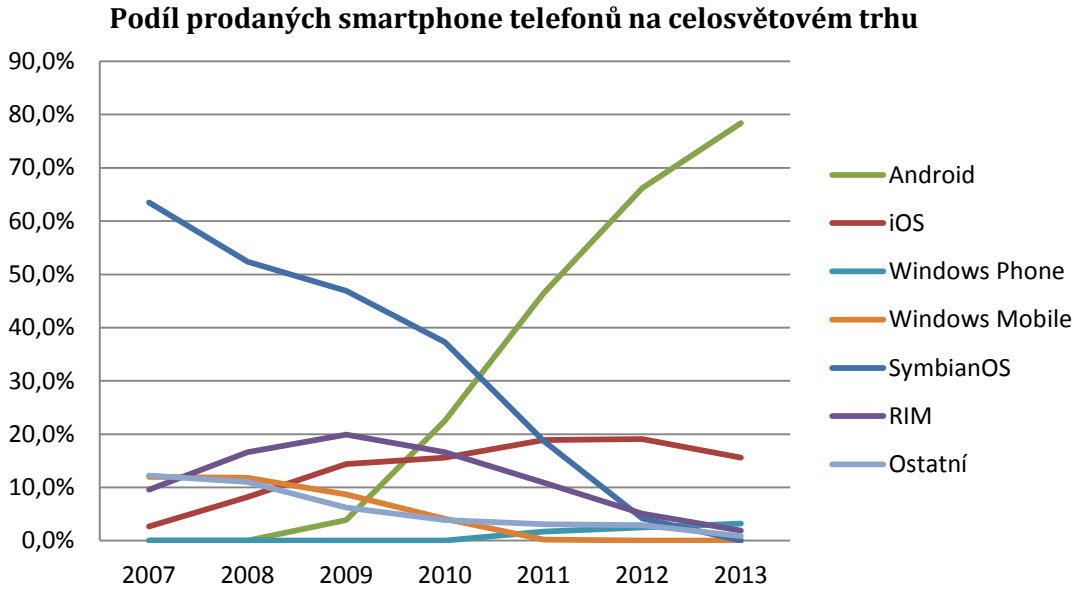
Při návrhu optimálního řešení pro mobilní platformy musíme vycházet ze základních faktů, a to že všechny zařízení se navzájem odlišují svým operačním systémem (Android, iOS, …) a svými rozměry a rozlišením obrazovky. Tato rozličnost často komplikuje proces návrhu mobilních aplikací.

Pro vývoj mobilních aplikací se používá několik základních principů, ke kterým patří například vývoj nativní aplikace nebo vývoj webové aplikace. Těchto přístupů je více a u všech je nutno zvážit jejich výhody a nevýhody a zvolit ideální postup vlastního řešení.

5.3.1 Nativní aplikace

Každý tablet nebo smartphone má vlastní operační systém, na kterém běží všechny jeho aplikace. K základním aplikacím většinou patří emailový klient, webový prohlížeč, mp3 přehrávač a další. Mobilní operační systémy umožnují instalovat i vlastní aplikace a rozšiřovat tím možnosti a využití zařízení. Tyto aplikace jsou nazývány nativními.

Nativní aplikace se vytváří přímo pro konkrétní mobilní platformu. Těch vznikla celá řada, ale jen několik málo z nich si získalo větší popularitu. K nejpoužívanějším operačním systémům se dnes řadí Android od společnosti Google (78%), iOS (15%) od společnosti Apple a určité zastoupení má i Microsoft s Windows Phone (3%). V předchozích letech měl většinové zastoupení systém Symbian od společnosti Nokia, ale ten už dnes zcela zanikl. Vývoj mobilních platforem se mění velmi rychle, protože jsou zatím v počátcích. Následující graf znázorňuje jejich vývoj od roku 2007.



Obrázek 5.6: *Graf vývoje prodejnosti smartphone na celosvětovém trhu. Zdroj dat* [19]

Každá z těchto platforem využívá svůj vlastní programovací jazyk a má jinou architekturu a vývojové prostředí pro vývoj aplikací. Díky nativnímu kódu lze pomocí aplikace přistupovat k hardwarovým funkcím telefonu a využívat možnosti fotoaparátu, GPS, gyroskopu, kalendáře a mnoha dalších.

Aplikace vyvinuté nativním způsobem jsou velmi rychlé a spolehlivé. Ke svému běhu a spuštění nevyžadují připojení k internetu a mohou běžet zcela offline. Díky optimalizaci kódu dokážou být i šetrné na výdrž baterie telefonu. Nativní aplikaci si uživatel může vyhledat a prodejním portálu dané platformy jako je App Store (iOS), Google Play (Android), Windows Store a nainstalovat přímo do zařízení. Snadná dostupnost aplikace na ploše telefonu je jednou z hlavních výhod nativního způsobu vytváření mobilních aplikací. Architektury jednotlivých platforem a jejich rozdíly při vývoji jsou popsány níže.

Android

Android je open-source platforma běžící na linuxovém jádře. Je určena pro mobilní zařízení typu Smart Phone. Společnost Android byla založena v roce 2003 Andym Rubinem a v roce 2005 ji odkoupila společnost Google. Od roku 2007 se Android stal standardem pro mnoho mobilních zařízení od společností jako je Samsung, HTC, Google nebo Motorola, kteří své telefony prodávají právě s touto platformou. Na trhu má proto Android dnes největší zastoupení.

Velké množství různých zařízení sebou nese ale i velkou nevýhodu s odlišným rozlišením obrazovek a různým výkonem zařízení. Aplikace je proto nutné vytvářet velmi univerzálně, aby podporovaly všechny typy zařízení.

Aplikace se vyvíjí v programovacím jazyce Java a doporučeným vývojovým prostředím je Eclipse. Pro programátory je určen vývojářský balík Android SDK (Software Development Kit), který obsahuje sadu knihoven, emulátor pro testování, dokumentaci i nástroj pro ladění chyb. Tento balík je volně k stažení na oficiálních stránkách. Android aplikace běží na virtuálním stroji zvaném Dalvik, který je na rozdíl od standartního Java Virtual Machine rychlejší a optimalizovaný přímo pro mobilní platformy. Linuxové jádro operačního systému navíc umožnuje souběžný běh aplikací tzv. multitasking. Vývojáři mohou přistupovat díky otevřenosti platformy ke všem systémovým a hardwarovým funkcím telefonu, což na druhou stranu snižuje její bezpečnost.

Distribuce android aplikací se provádí přes tzv. Google Play. Tato služba je dostupná na internetu a umožňuje vyhledávání a stahování aplikací přímo do telefonu. Při prodeji aplikací je nutno se na marketu zaregistrovat a zaplatit jednorázový poplatek 25 dolarů.

iOS

Operační systém iOS vytvořila společnost Apple. Platforma vychází ze systému Mac OS a je určena pouze pro zařízení společnosti Apple (pro iPhone, iPad, Apple TV a další). Pro vývoj aplikací slouží vývojové prostředí XCode. Toto prostředí je dostupné pouze pro operační systém Mac OS, což znemožňuje vývoj ve Windows nebo Linuxu. Systém je založený na unixovém jádře a dělí se do několika vrstev, které svým vývojářům poskytují aplikační rozhraní i rámce potřebné k vývoji aplikací.

Jedná se o uzavřenou platformu, která svým uživatelům znemožňuje přístup do systému. Aplikace mohou být nainstalovány pouze přes oficiální on-line obchod zvaný App Store, kde aplikace musí projít schvalovacím procesem. Všechny tyto opatření vedou k zvýšené bezpečnosti systému a k zamezení jeho napadení a zneužití.

Při prodeji vlastních aplikací na appStore se musí vývojář zaregistrovat a zaplatit roční poplatek 99 dolarů. Vývoj je u iOS nákladnější a složitější než v případě Android platformy, ale zato výnosnější.

Windows Phone

Windows Phone je operační systém, který vyvinula společnost Microsoft. Tato platforma byla vydána v roce 2010 a zcela nahradila starší operační systém Windows Mobile. Je určen pro chytré mobilní telefony a svým uživatelům nabízí nové uživatelské rozhraní zvané Metro s dlaždicovým uspořádáním.

V současné době je nejaktuálnější verzí systému Windows Phone 8, který má zcela nové jádro systému zvané Windows NT. Přechod na nové jádro způsobil, že zařízení s aktuálním systémem není kompatibilní s předchozí verzí Windows Phone 7.

Architektura aplikací je stavěna na systémové knihovně Windows Runtime (Windows RT), která umožnuje aplikace vyvíjet v několika programovacích jazycích, k nimž patří C#, Visual Basic, C++ nebo XAML. Vývojáři si tak mohou zvolit jazyk, který zvládají nejlépe a nemusí se učit nic nového. Pro vývoj je nutné si nainstalovat operační systém Windows 8 a vývojové prostředí Visual Studio 2012 nebo jeho zjednodušenou verzi Visual Studio Express, která je dostupná i zdarma.

Předním výrobcem zařízení s operačním systémem Windows 8 se stala společnost Nokia, k dalším pak patří společnost HTC, Samsung, Acer nebo LG.

K distribuci aplikací vyvinutých pro Windows 8 slouží Windows Store, který je obdobou Apple Store a umožnuje za určitý poplatek prodávat a stahovat hotové aplikace.[15]

Shrnutí vývoje nativních aplikací

Nativní aplikace dokážou být velmi výkonné a využívat všechny funkce telefonu ať už zvolíme jakýkoliv operační systém. Na druhou stranu tyto aplikace postrádají přenositelnost a nelze je spustit na zařízení s jinou mobilní platformou. Pro každý operační systém je nutné vytvářet nativní aplikaci zvlášť, což může být časově i finančně náročné. Z tohoto hlediska je výhodnější zvolit jiný způsob tvorby aplikací a to např. vývoj webové aplikace.

5.3.2 Webové aplikace

Webová mobilní aplikace má podobu dynamických webových stránek, jejichž obsah je přizpůsoben malým velikostem obrazovek a dotykovému ovládání telefonů. Je psána pomocí jazyka HTML, CSS a Javascript. Webová aplikace se nemusí nijak instalovat a spouští se pouze pomocí mobilního webového prohlížeče, tudíž šetří vnitřní paměť zařízení. Výhodou webové aplikace je již zmíněná přenositelnost. Stačí ji jednou vytvořit a lze ji pak spouštět v jakémkoliv zařízení nezávisle na jeho platformě. Jazyk HTML 5 zčásti eliminuje původní nedostatky webu a dokáže využívat i některé hardwarové funkce mobilních telefonů.

Nevýhodou webové aplikace je ale nutnost připojení k internetu, jehož rychlostí bývá ovlivněna i rychlost samotné aplikace. HTML 5 se snaží tento nedostatek zčásti eliminovat použitím interní cache a lokálním ukládáním stránek. To umožnuje uživatelům zobrazovat již navštívené stránky a eliminuje ztrátu dat při výpadku internetu (např. emailová zpráva). Data

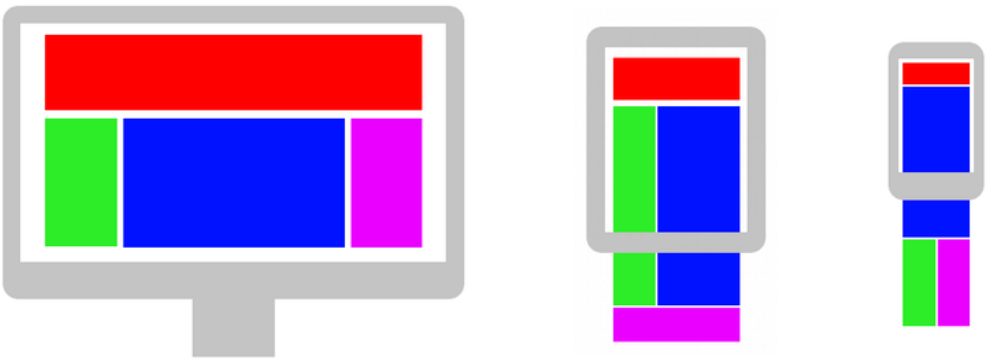
jsou lokálně uložena a synchronizována se serverem až když je připojení k internetu k dispozici. Tímto je možno vytvářet stabilnější aplikace i při nestabilním připojení k internetu. [15]

5.3.3 Hybridní aplikace

Hybridní aplikace spojuje výhody obou předchozích řešení. Jedná o webovou aplikaci vytvořenou pomocí HTML, CSS a Javascript knihovny zabalenou do nativní aplikace pomocí speciálních emulátorů jako je PhoneGap. Výsledkem emulátorů bývají finální aplikace pro iOS, Android a Windows Phone. Hybridní aplikace tak může být nainstalována na všechny platformy bez nutnosti znát jejich vývojové prostředí a programovací jazyk. Pro jednoduché aplikace je proto tento hybridní vývoj nejvhodnějším řešením.

5.3.4 Responzivní design (mobilní šablona webu)

Responzivní design je pojem, jehož autorem je americký programátor Ethan Marcotte. Jedná se o úpravu webových stránek do podoby, která je čitelná a uživatelsky přívětivá i pro mobilní zařízení. Webová stránka se pouze upravuje kaskádovými styly (CSS) a optimalizuje do výsledného řešení. Jedná se o stejnou instanci stránky, jakou si uživatel zobrazuje v prohlížeči na svém počítači na rozdíl od webové aplikace, kdy se vytváří samostatná oddělená verze stránky přizpůsobená mobilním telefonům na míru.



Obrázek 5.7: *Příklad zobrazení webové stránky s responzivním designem* [16]

Stránka s responzivním web designem se skládá z plovoucích prvků, které se přizpůsobují velikosti a orientaci obrazovky. Na obrázku 5.7 lze vidět příklad rozvržení stránky pro monitor, tablet a smartphone. Flexibilní struktury se dosahuje pomocí procentuálního vyjádření šířky elementů a obrázků a s využitím tzv. *Media Queries* vlastností, které jsou zahrnuty v specifikaci CSS3. Media Queries jsou pravidla, která jsou aplikovaná na styl dokumentu v závislosti na velikosti zobrazovaného zařízení.

K výhodám tohoto řešení je, že je nejméně náročné na úpravu pokud má původní řešení formu webové stránky. Oproti předchozím principům ale nepodporuje offline mód, nemá přístup k hardware funkcím přístroje a nedokáže využívat pohybová gesta na obrazovce. [14]

5.3.5 Zhodnocení a návrh vlastního řešení

Pro lepší srovnání všech zmíněných principů a jejich nabízených funkcí slouží následující tabulka.

	Mobilní aplikace	**Webová aplikace**	**Hybridní aplikace**	**Responzivní šablona**
Přístupnost řešení skrz všechny platformy	✖	✔	✔	✔
Přístup k hardwarovým funkcím telefonu	✔	✔ (částečně)	✔ (částečně)	✖
Využití dotykových gest	✔	✔	✔	✖
Aplikace je k dispozici bez instalace	✖	✔	✖	✔
Vysoká uživatelská přívětivost	✔	✖	✖	✖
Využití aplikace bez připojení k internetu (offline)	✔	✔ (částečně)	✔ (částečně)	✖
Jednoduchý a rychlý vývoj bez potřeby znalostí objektově orientovaných jazyků (C#, Java, Objective-C) a jejich vývojových prostředí.	✖	✔	✔	✔
Okamžité nasazení aplikace a aktualizace bez nutnosti schvalovacího procesu	✖	✔	✔	✔

Tabulka 5.1: *Tabulka srovnání různých principů při tvorbě mobilní aplikace*

Po shrnutí všech výhod a nevýhod nelze říct, že jeden z těchto principů vývoje je daleko lepší než ostatní. Rozhodnutí závisí hlavně na požadavcích konkrétního řešení. V mém případě jsou tyto požadavky na výslednou aplikaci následující:

- Aplikace bude využívat připojení k internetu – pro stahování testů, autentizaci uživatele a synchronizaci jeho výsledků
- Nebude přistupovat k žádným hardwarovým funkcím telefonu
- Nepotřebuje využívat dotyková gesta
- Nasazení aplikace by mělo být okamžité bez schvalovacích procesů
- Vývoj by měl být co nejrychlejší a nejefektivnější

Při shodnocení všech požadavků jsem dospěla k závěru, že by výsledné řešení mělo mít podobu webové aplikace nebo responzivní šablony. Vytváření nativní aplikace by bylo příliš časově náročné a navíc by výsledné řešení nebylo multiplatformní. Dosavadní systém již využívá webové standarty, a tak je nejefektivnější ji převést a optimalizovat do mobilní webové podoby.

5.3.6 Návrh mobilní aplikace v prostředí ASP.NET

Vlastní řešení systému je psáno v jazyce ASP.NET Web Forms. Adaptace systému pro mobilní platformy se bude týkat pouze prezentační části systému, protože pouze v ní bude mít nejvyšší užitnou hodnotu. Pro lektora je pohodlnější a praktičtější vytvářet kurzy na počítači, a proto je zbytečné jeho část systému optimalizovat taktéž.

Microsoft svým vývojářům pro ASP.NET nabízí dva způsoby, jak jejich řešení přizpůsobit mobilním platformám. Buď tuto problematiku adaptace vyřešit na straně klienta pomocí CSS stylů a Media Query (princip responzivní šablony) nebo na straně serveru (webová aplikace). Jelikož mé řešení pro mobilní platformy se bude lišit od původního řešení v nabízených funkcích, bude se lišit i HTML výstup aplikace. Zásadnější změny designu by se pouze pomocí kaskádových stylů dělaly velmi obtížně, proto je lepší zvolit druhý přístup tvorby aplikace na straně serveru. Nicméně oba přístupy se navzájem nevylučují, a tak je možné je zkombinovat dohromady a responzivní design používat pro doplnění výsledné mobilní webové aplikace.

Princip vývoje webové aplikace v prostředí ASP.NET je založen na rozpoznání mobilního zařízení. V závislosti na identifikaci telefonu lze v programu měnit vzhled i logiku webových stránek. Může tak být vytvořena zcela odlišná instance webu pro mobilní zařízení. [17]

Detekce zařízení a prohlížeče

Jedním z hlavním klíčových předpokladů pro tvorbu mobilních aplikací je znalost typu prohlížeče a mobilního zařízení, které návštěvník používá. K detekci těchto charakteristik slouží v ASP.NET následující vlastnosti třídy *Request.Browser*:

- *Request.Browser.IsMobileDevice*

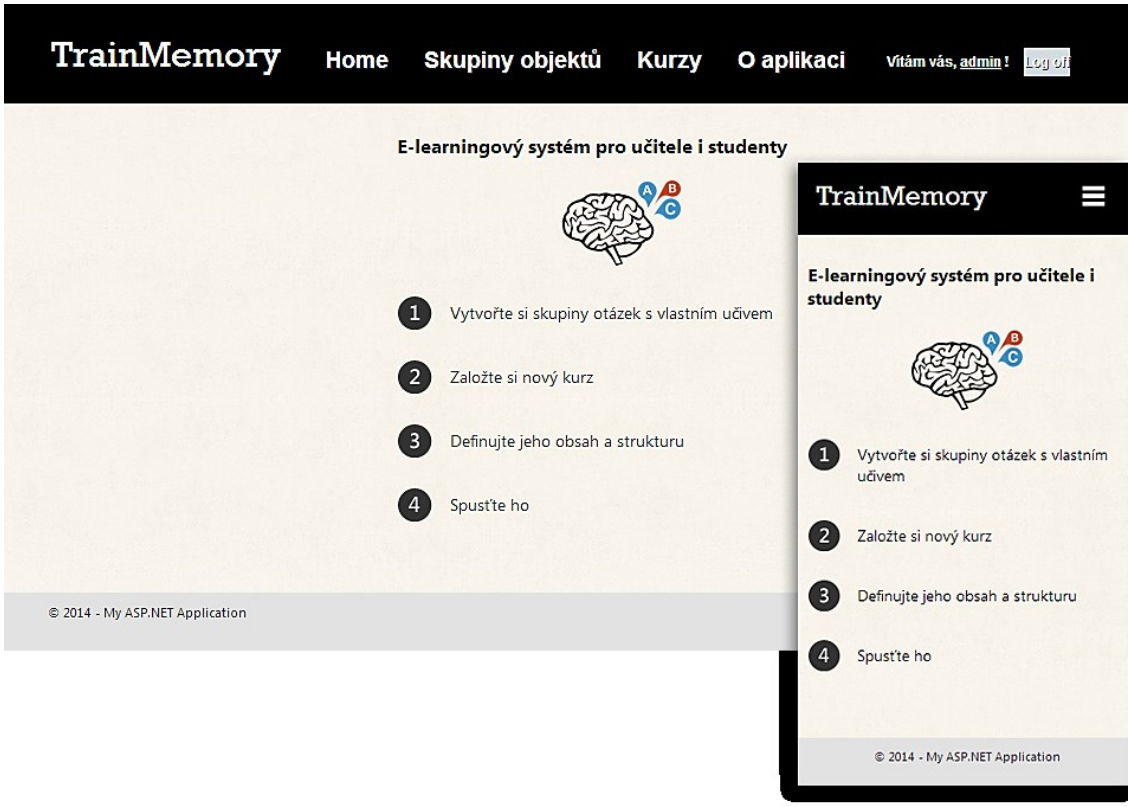
- *Request.Browser.MobileDeviceManufacturer, Request.Browser.MobileDeviceModel*
- *Request.Browser.ScreenPixelsWidth*
- *Request.Browser.SupportsXmlHttp*

```
protected void Page_Init(object sender, EventArgs e)
{
    if (Request.Browser.IsMobileDevice){
        mobilemenu.Visible = true;
        …
   }
}
```

Kód znázorněný výše je použit v projektu. V případě rozpoznání mobilního zařízení zobrazí jinou komponentu menu s odlišnými záložkami určenými pouze pro spouštění kurzů (ne pro vytváření). Stejným způsobem lze měnit i celé rozležení webu (Site.Master) a jakékoliv jiné části aplikace.

Použití responzivního designu

Pro výsledné řešení byla použita i technika responzivního designu a jeho speciálních CSS3 stylů. Styly jsou dynamicky měněny v závislosti na velikosti obrazovky. Na následujícím obrázku lze vidět výsledné řešení při standartním rozlišení monitoru a při rozlišení 320 x 480px.



Obrázek 5.8: *Responzivní design aplikovaný v systému*

Pro dosažení změn v hlavičce stránky byly použity tagy *Media Queries* a jQuery knihovna *Responsive-menu*, která zajistila rozbalování menu.

Speciální styly byly aplikovány na velikost obrazovky s šířkou menší než 600px, kdy se položky menu začínaly řadit pod sebe a menu se rozpínalo do výšky. Podmíněná syntaxe pro určování stylů pro menší zařízení vypadá následovně:

```
@media only screen and (max-width : 600px) {
   .header_menu {
         float: right;
    }
}
```

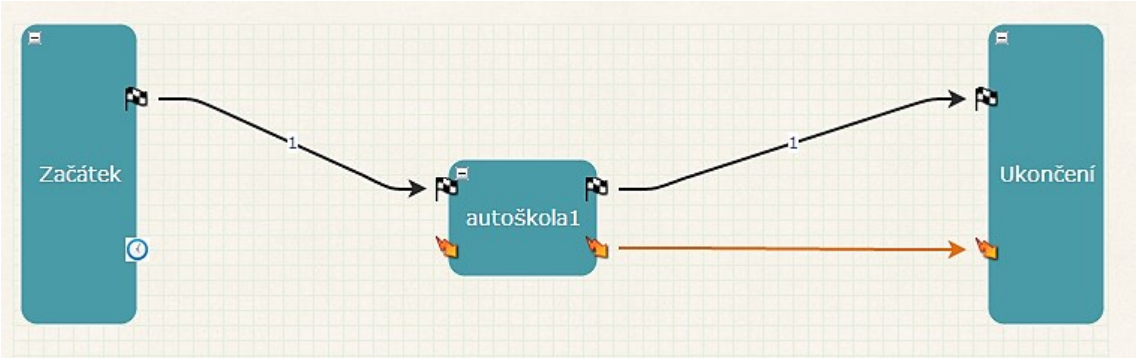
S kombinací responzivního designu a ASP.NET funkcí pro detekci mobilního zařízení byl vytvořen layout systému optimalizovaný pro mobilní zařízení. Funkce systému pro testování je proto možné využívat i na mobilních platformách bez omezení.

6 Praktické ověření funkčnosti

Ověření funkčnosti systému bylo provedeno na několika cvičných kurzech. Byly vytvořeny tři kurzy s odlišnou strukturou, které jsou u lektorů často využívané.

První kurzem je **lekce Autoškoly**. Jedná se jednoduchý kurz, kdy jsou uživateli zobrazovány otázky za sebou v časové souslednosti. Student odpovídá na otázku výběrem z více možností odpovědí. Na konci testu je zobrazen výsledek kurzu.

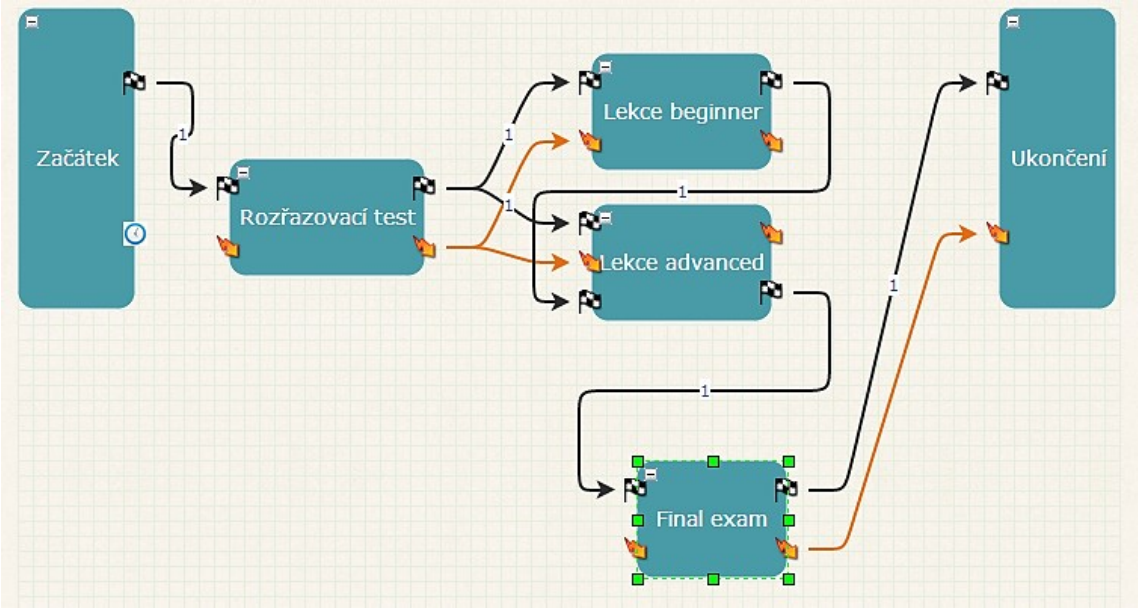
Při návrhu řešení kurzu s natolik jednoduchou strukturou nebyl systém příliš efektivní. Uživatel, který kurz vytvářel, musel nejdříve vytvořit skupinu otázek s příklady. Poté musel vytvořit kurz, přiřadit k němu skupinu, nastavit ji a určit strukturu kurzu, která v tomto případě vypadala následovně.



Obrázek 6.1: *Příklad kurzu Autoškoly*

Při spouštění pouze jedné skupiny otázek je vytváření struktury kurzu zbytečně zdlouhavé. Nicméně struktura kurzu je navržena tak, aby byla co nejvíce univerzální, což je bohužel na úkor jednoduchosti. Tento problém by možná šel vyřešit v budoucnu možností spouštět samotné skupiny otázek. Pro potřeby odchytávání výsledků ze skupin otázek je nutno pak upravit i databázi systému.

Dalším typem kurzu je například **kurz Angličtiny**, obsahující 2 lekce a 2 testy zobrazené na obrázku 6.2. Uživateli je nejdříve zobrazen rozřazovací test, na jehož základě jsou zobrazeny následující lekce. Kurz obsahuje adaptivní prvky, neboť volba následující lekce závisí na výsledku z testu. Lektor nejdříve vytváří 4 skupiny otázek. První dvě skupiny zvané *Lekce Beginner* a *Lekce Advanced* obsahují pouze výkladové části, zatímco skupiny s názvem *Rozřazovací test* a *Final exam* budou obsahovat pouze testovací otázky. Struktura kurzu bude vypadat následovně.



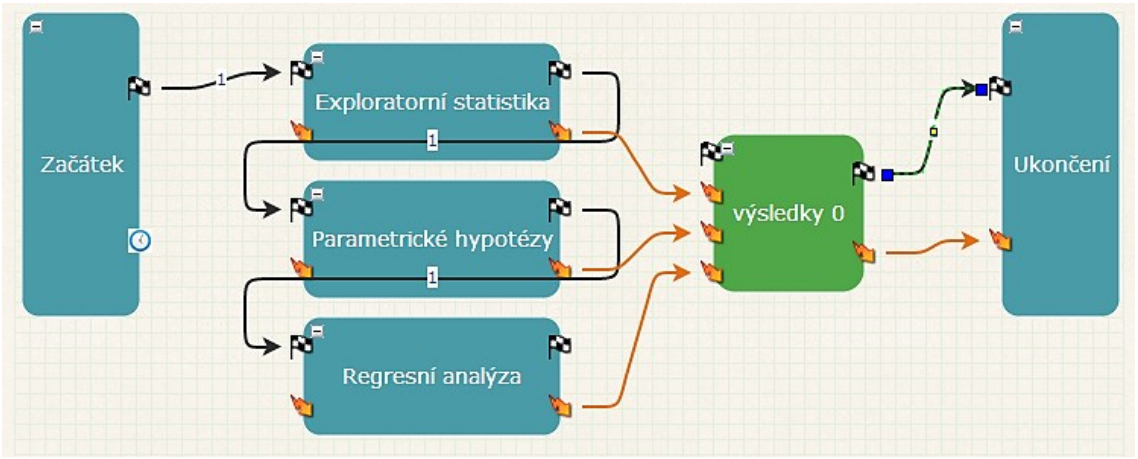
Obrázek 6.2: *Příklad kurzu Angličtiny*

Struktura kurzu Angličtiny má oproti předchozímu kurzu složitější strukturu. Na vstupech modulů *Lekce beginner* a *Lekce advanced* byla umístěna vstupní podmínka, která udávala procentuální hranici dosažených bodů, které student musí dosáhnout, aby byl modul spuštěn. Tvar podmínky byl následující.

```
[VAR8 - Rozřazovací test]>50          .. v případě lekce advanced
[VAR8 - Rozřazovací test]<=50         .. v případě lekce beginner
```

Navržená struktura kurzu byla otestována jejím spuštěním. V prvním případě, když byly všechny otázky v rozřazovacím testu zodpovězeny špatně, byl uživateli následně zobrazen modul pro začátečníky (Lekce beginner). Průběh kurzu měl pořadí *Rozřazovací test* → *Lekce Beginner* → *Lekce Advanced* → *Final exam*. V druhé instanci vykonávání kurzu byly mnohé otázky zodpovězeny správně a výsledek byl 75%. Průběh kurzu měl proto jiné pořadí*:* *Rozřazovací test* → *Lekce advanced* → *Final exam*.

Posledním příkladem kurzu bude **kurz Statistiky**, jehož struktura je znázorněna na dalším obrázku.



Obrázek 6.3: *Příklad kurzu Angličtiny*

Požadavkem bylo, aby finální výsledek kurzu mohl být ovlivněn lektorem. V kurzu statistiky jsou 3 lekce *Exploratorní statistika*, *Parametrické hypotézy* a *Regresní analýza*. Největší důraz klade lektor na znalosti *Regresní analýzy*, a proto by chtěl, aby se výsledek z tohoto modulu více promítl do finálního ohodnocení.

K tomuto účelu slouží výsledkový typ modulu (zelený), který je schopen definovat nové proměnné na základě přijatých hodnot. Každý testovací modul běžně vrací výsledek v procentuálním váhovém průměru. Zvýhodnění jednoho výsledku může probíhat způsobem, že ho vynásobíme jiným koeficientem než ostatní proměnné.

Algoritmus pro výpočet proměnné zadávané do vstupního může mít následující tvar:

```
(2/4*[var14 - Regresní analýza]+1/4*[var15 - Parametrické
hypotézy]+1/4*[var16 - Exploratorní statistika])/3
```

Jedná se o klasický aritmetický průměr zkombinovaný s vlastními koeficienty. Lektor při jeho vytváření zvolil přímý způsob definování proměnné a do vstupního pole pomocí kláves s proměnnými vyplnil algoritmus. Lektorovi tak mohlo být vyhověno v jeho požadavcích a studenti museli lépe zvládnout regresní analýzu, aby dosáhli vyššího skóre.

Při testování se zjistilo ale pár drobných nedostatků z hlediska navrženého designu. Například při definici příliš dlouhé podmínky nebo algoritmu, bylo pole pro vstupní řetězec příliš malé. Také při tvorbě obsáhlejších kurzů složených z mnoha modulů byla malá obrazovka designeru nevýhodou a struktura kurzu byla špatně čitelná.

Testování systému probíhalo i s dalšími účastníky, jejichž úkolem bylo vytvořit různé kurzy. Účastníci měli problém zpočátku se v systému vyznat a pochopit jeho strukturu. Po krátkém vysvětlení byli schopni se systémem pracovat na základní úrovni. Dokázali vytvářet skupiny otázek a vkládat je do kurzu. Z designeru pro vytváření struktury kurzu byli zpočátku zmatení. Později však ocenili intuitivní ovládání a přehlednost při tvorbě kurzů. Poté jim byla nabídnuta mobilní forma systému pro testování, s jejichž ovládáním nikdo problém neměl.

Vzhledem k tomu, že nástroj pro vytváření kurzu je určen zejména pro lektory, je nutné, aby vytváření kurzu bylo rychlé a přehledné i na úkor horší počáteční orientace v systému. Systém je navržen tak, aby lektory nezatěžoval vyplňováním zbytečných vstupních polí nebo zobrazováním obrazovek, které zrovna nepotřebují.

7 Závěr

7.1 Zhodnocení dosažených cílů práce

Diplomová práce si kladla za cíl navrhnout nástroj pro vytváření e-learnigových kurzů spustitelných v mobilních zařízeních. Cíle bylo dosaženo splněním dílčích požadavků zmíněných v kapitole 4.2.

Ve výsledku vznikl systém, který má podobu webové aplikace navržené v prostředí ASP.NET. Systém je schopen navrhovat libovolné struktury kurzů, jak bylo požadováno v zadání. Navržená struktura kurzu je schopna pojmout výukové materiály s libovolným způsobem jejich procházení a je podrobně popsána v kapitole 4.5.

Hierarchie kurzu byla rozdělena do čtyř nezávislých celků odpovídající komponentě výukový objekt, skupina objektů, modul a kurz. Nejnižší komponentu pro uchovávání výukových materiálů (skupinu objektů) je možno zahrnout i do jiných kurzů, čímž bylo dodrženo pravidlo znuvupoužitelnosti.

Při nastudování problematiky adaptivních kurzů, byly i do vlastního řešení zahrnuty prvky adaptivního průchodu skrz kurzem a funkce podmíněného vykonávání aktivit. Logika testování podporující adaptivní průchod byla navržena s využitím reflexe a je popsána v kapitole 5.1.3.

Druhou část systému měla tvořit prezentační vrstva, ve které mělo docházet k testování a spouštění kurzů. Jedním z hlavních požadavků bylo, aby vrstva byla optimalizována pro mobilní platformy a zkoušení mohlo tudíž probíhat na mobilním zařízení typu smartphone. Volbou vhodného způsobu implementace prezentační části přizpůsobeného mobilním platformám se zabývá kapitola 5.3. Při zhodnocení všech postupů byl zvolen princip webové aplikace s využitím responzivního designu. Zvolené řešení umožnuje výukový systém spustit na jakémkoliv chytrém mobilním telefonu nezávisle na jeho operačním systému.

Realizovaný systém nenabízí tolik funkcionalit jako například LMS Moodle nebo Adobe Captivate. Jeho výhoda zato tkví v univerzálnosti struktury, díky níž může být systém snadno rozšířen o další typy výukových objektů (úloh) nebo modulů, a svým uživatelům tak nabízet širší využití i v budoucnosti.

7.2 Možnosti budoucího vývoje

Systém má mnoho možností, jakými může být v budoucnu rozšířen. Aktuální stav umožnuje vytváření různorodých kurzů, ale jejich různorodost by mohla být ještě větší s přidanými novými typy otázek nebo modulů. Také možnosti sdílení kurzu mezi uživateli a nastavování přístupových práv by bylo vhodné začlenit do budoucího řešení. Široký rozsah práce už příliš neumožnoval zabývat se analýzou získaných výsledků z kurzů a využívat je pro zefektivnění výukového procesu. I tímto směrem by se mohla práce v budoucnu rozvíjet. Budoucí vývoj systému by mohl řešit níže uvedené problematiky:

- Navržení a přidání nových typů úloh jako je doplňovací úloha, přiřazovací úloha, hot spot apod.
- Navržení a implementace nových typů modulů, které budou rozšiřovat funkce systému při vytváření kurzu. Novým typem modulu by mohl být například modul se vstupním polem pro zadávání proměnných přímo od uživatele. Uživatel vykonávající kurz by tak mohl vyplnit například svůj věk, jehož hodnota by mohla být využita při vyhodnocení kurzu (IQ testy).
- Sdílení kurzů a nastavování přístupových práv
- Zpracování a využití výsledků uživatele pro tvorbu statistik a zefektivnění výukového procesu

Použitá literatura

[1] FELDSTEIN, Michael. What's important in a learning content management system. [online]. [cit. 2014-04-20]. Dostupné z: *http://elearnmag.acm.org/featured.cfm?aid=566792*

[2] KERSCHENBAUM, Steven a Barbara T. WISNIEWSKI BIEHN. LMS Selection Best Practices. [online]. [cit. 2014-04-20]. Dostupné z: *http://www.trainingindustry.com/media/2068137/lmsselection_full.pdf*

[3] The Campus Computing Project [online]. 2011 [cit. 2014-04-21]. Dostupné z: *http://www.campuscomputing.net/sites/www.campuscomputing.net/files/Green-CampusComputing2011_5.pdf*

[4] SCORM. [online]. [cit. 2014-04-20]. Dostupné z: *http://scorm.com/*

[5] SCORM 2004 - PŘEHLED: KONFERENCE BELCOM'05 TRENDY V E-LEARNINGU [online]. [cit. 2014-04-21].

[6] ADLnet. [online]. [cit. 2014-04-21]. Dostupné z: *http://www.adlnet.org*

[7] HOLUB, Libor. Automatizované řízení adaptivní výuky v e-learningu podle stylů učení studenta Studijní. Ostrava, 2010. Disertační práce. VŠB – Technická univerzita Ostrava.

[8] ŠARMANOVÁ, Jana a Kateřina KOSTOLÁNYOVÁ. Tvorba adaptivních e-learningových opor: metodická příručka pro autory. Ostrava, 2011. Skripta. Ostravská univerzita.

[9] VOTAVA, Jiří. STRATEGIE UČENÍ [online]. [cit. 2014-04-21]. Dostupné z: *http://etext.czu.cz/img/skripta/64/pef_2314-1.pdf*. Skripta.

[10] Conditional activities settings. [online]. [cit. 2014-04-21]. Dostupné z: *http://docs.moodle.org/24/en/Conditional_activities_settings*

[11] Adobe Captivate Quiz Result Analyzer. In: [online]. [cit. 2014-04-21]. Dostupné z: *http://blogs.adobe.com/captivate/2010/08/adobe-captivate-quiz-result-analyzer.html*

[12] Adobe Captivate. [online]. [cit. 2014-04-21]. Dostupné z: *http://www.adobe.com/cz/products/captivate/educationsoftware.html*

[13] CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. Vytváříme mobilní web a aplikace pro chytré telefony a tablety. 1. vyd. Brno: Computer Press, 2013, 288 s. ISBN 978-80-251-3763-5.

[14] MARCOTTE, Ethan. Responsive Web Design. [online]. [cit. 2014-04-19]. Dostupné z: *http://alistapart.com/article/responsive-web-design*

[15] TOKÁR, Daniel. Nativní vs. webový vývoj aplikací na mobilní platformy. Brno, 2011. Dostupné z: *http://is.muni.cz/th/325070/fi_b/bc.pdf*. Bakalářská práce. Masarykova univerzita Brno. Vedoucí práce RNDr. Jaroslav Škrabálek.

[16] PROCHÁZKA, Tomáš. Jedno z možných řešení, jak rozdělit a zobrazovat obsah při použití responzivního web designu pro různé druhy zařízení. In: [online]. [cit. 2014-04-19]. Dostupné z: *http://commons.wikimedia.org/wiki/File%3AComplete.png*

[17] How To: Add Mobile Pages to Your ASP.NET Web Forms / MVC Application. [online]. [cit. 2014-04-22]. Dostupné z: *http://www.asp.net/whitepapers/add-mobile-pages-to-your-aspnet-web-forms-mvc-application*

[18] ŽÓŁTÁ, Veronika. Nástroj pro sebetestování se speciálními typy testovacích úloh. Ostrava, 2011.

[19] Gartner Inc. In: [online]. [cit. 2014-04-25]. Dostupné z: *http://www.gartner.com*

Seznam příloh

Příloha A: *Obsah připojeného CD*

Adresář	**Obsah**
/…	vypracovaná práce ve formátu PDF
/…	zadání diplomové práce ve formátu PDF
/ implementace / Application /	zdrojové kódy systému
/ implementace / Database /	export SQL databáze

Příloha B: *Kompletní datový model*